2013-06

UAV swarm tactics: an agent-based simulation and Markov process analysis

Gaerther, Uwe

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UAV SWARM TACTICS: AN AGENT-BASED SIMULATION AND MARKOV PROCESS ANALYSIS

by

Uwe Gaertner

June 2013

Thesis Advisor:	Timothy H. Chung
Second Reader:	Michael Atkinson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
20–6–2013	Master's Thesis	2011-06-24–2013-06-21

4. TITLE AND SUBTITLE

UAV SWARM TACTICS: AN AGENT-BASED SIMULATION AND MARKOV PROCESS ANALYSIS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Uwe Gaertner

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

The rapid increase in the use of unmanned aerial vehicles (UAVs) in recent decades lead to their potential use as saturation or swarm threats to Allied Forces. One possible counter measure is the design and deployment of a defensive UAV swarm. This thesis identifies a future concept of swarm-versus-swarm UAV combat, focusing on the implications of swarm tactics and identifies important factors for such engagements. This work provides initial key insights through significant modeling, simulation, and analysis. The contributions of the presented work include the design of an agent-based simulation and the formulation of an associated analytical model. The agent-based simulation allows for the UAV to be modeled as an agent that follows a simple rule set, which is responsible for the emergent swarm behavior relevant to defining swarm tactics. A two-level Markov process is developed to model the air-to-air engagements, where the first level focuses on one-on-one combat while the second level incorporates the results from the first and explores multi-UAV engagements. Tactical insights obtained from this study can be contrasted with tactics for manned air combat, which highlights the potential need to develop new tactics for unmanned combat aviation as well as for swarm scenarios. Additional analysis performed in this thesis provides further tactical recommendations and outlines multiple avenues of future study.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	119	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UAV SWARM TACTICS: AN AGENT-BASED SIMULATION AND MARKOV PROCESS ANALYSIS

Uwe Gaertner
Captain, German Army
Diplom, University of the German Armed Forces Munich, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2013

Author: Uwe Gaertner

Approved by: Timothy H. Chung
Thesis Advisor

Michael Atkinson
Second Reader

Robert F. Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The rapid increase in the use of unmanned aerial vehicles (UAVs) in recent decades lead to their potential use as saturation or swarm threats to Allied Forces. One possible counter measure is the design and deployment of a defensive UAV swarm. This thesis identifies a future concept of swarm-versus-swarm UAV combat, focusing on the implications of swarm tactics and identifies important factors for such engagements. This work provides initial key insights through significant modeling, simulation, and analysis. The contributions of the presented work include the design of an agent-based simulation and the formulation of an associated analytical model. The agent-based simulation allows for the UAV to be modeled as an agent that follows a simple rule set, which is responsible for the emergent swarm behavior relevant to defining swarm tactics. A two-level Markov process is developed to model the air-to-air engagements, where the first level focuses on one-on-one combat while the second level incorporates the results from the first and explores multi-UAV engagements. Tactical insights obtained from this study can be contrasted with tactics for manned air combat, which highlights the potential need to develop new tactics for unmanned combat aviation as well as for swarm scenarios. Additional analysis performed in this thesis provides further tactical recommendations and outlines multiple avenues of future study.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ABS Agent-Based Simulation
ABM Agent-Based Model
API Application Programming Interface
CCD Center Composite Design
COM Center Of Mass
CRN Common Random Numbers
CSV Comma Separated Values
DoE Design of Experiment
GLM Generalized Linear Model
HVT High Value Target
JAR Java ARchive
JMF Java Media Framework
JRE Java runtime environment
Mason Multi-Agent Simulator Of Networks
MOE Measure Of Effectiveness
MOP Measures Of Performance
NOLH Nearly Orthogonal Hypercube Design
NPS Naval Postgraduate School
PRAWN PRoliferated Autonomous WeapoN
SO Self-Organization
UAV Unmanned Aerial Vehicle
USG United States Government

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

With the advent of unmanned combat aerial vehicles, the present day may represent one of the most significant periods of change in the battle space since the first use of aircraft in warfare. Unmanned aerial vehicles (UAVs) are increasingly tasked with more missions, including those originally done by manned fighter aircraft. The advantages are obvious: first of all, they save lives because pilots are not exposed in the battle space. UAVs also have the advantage (either presently or in the future) of being cheaper in procurement, operation, maintenance, and necessary ground personnel. Such benefits do not stop at simply replacing manned aircraft with UAVs. The potential applications are even more significant, such as using autonomous swarms of UAVs as the next evolution of aerial warfare. The drawback of these emerging technologies is that potential adversaries also recognize these advantages of unmanned systems, which can then pose a threat to allied forces, including saturation attacks with large numbers of weapons and/or unmanned systems. One possible countermeasure to such threats is a defensive UAV swarm.

The concept of air-to-air combat with unmanned combat aerial vehicles is still in its nascent stages, and this work proposes and explores a novel future concept in which swarms of UAVs combat the adversary's UAV swarm. Though there is substantial scientific literature available which address aspects of UAV swarms such as self-organization, UAV system measures, or multi-UAV search and detection approaches, this thesis uniquely investigates the development of *tactics* specifically addressing swarm versus swarm engagements. Even if tactics in manned air-to-air combat have previously been discussed during the last century of naval aviation, we assume that the employment of UAV swarms distinctly offers new tactics and merits revision of existing ones. The development of these *swarm tactics* motivates this thesis, with the goal of identifying influential factors and providing a foundation for follow-on research.

First we define a simple scenario that drives the presented work, motivated by ongoing proof-of-concept. We then develop an agent-based simulation in a bottom-up process, in which each UAV is treated as an agent in a network that forms the swarm. The agent is endowed with a small set of rules concerning the agent's motion, combat behavior, and its interaction with teammates. The behavior set itself can be extended to the needs of other research questions in swarm vs. swarm engagements. By construction, there is no central controller that manages the agents; rather, all UAVs are assumed to be vehicles acting autonomously. Design of the agents' rule sets leads to emergent behaviors of the collective, which are observed in the simulation

and can be evaluated across varying input parameters defined in this thesis. Statistical design of experiments in the form of a central composite design is used to scan the factor space for the identified parameters. Among other parameters, we vary the initial positioning of a swarm, the weighting factor between preferences of the swarm for offensive and defensive behavior, and the maximum number of assignments of friendly UAVs to each detected enemy UAV. Logistic regression on the responses of the Monte Carlo simulation runs show that all identified factors are nominally important to explain the model behavior, though the weight factors for each respective swarm are determined to be the most significant ones. Though the presented scenario explores fixed swarm sizes of 50 versus 50 UAVs, the analysis provides insights on the effectiveness of a smaller number of UAVs on one side. For example, such a smaller swarm can be seen to successfully destroy the adversary's high value unit by concentration of force with a high preference for offensive attack. We see that such behavior incurs substantial losses; however, for sufficiently expendable unmanned assets, the potential operational and tactical advantages of destroying a valuable target may outweigh such losses.

Another approach explored in this thesis is an analytic model formulation to describe the swarm versus swarm engagement. A two-level Markov process first looks at one-on-one UAV combat, which then incorporates the resulting insights in the transition probabilities into an expanded model of the multi-UAV battle. Sensitivity analysis on this theoretical formulation shows similar results as provided by the agent-based simulation model, serving as reasonable verification of the models. Continued theoretical analysis is thus determined to be a promising avenue of future work, in which the Markov process model can provide an efficient analysis tool to generate high-level insights, and coupled with the higher-fidelity agent-based simulation model, can robustly explore the space of swarm tactics.

The presented work additionally highlights many potential opportunities for further research, and provides a foundation to encourage other researchers to further explore this interesting and emerging capability area.

Acknowledgements

I would like to thank my advisor Timothy Chung for giving me the opportunity to explore this interesting research area and for always encouraging me to learn more about this interesting topic. His enthusiastic and positive nature made it an enjoyable lesson for me. I also thank Lyn Whitaker who walked me through the analysis and spent many hours to teach me advanced logistic regression techniques.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Background

Unmanned aerial vehicles (UAVs) have played an increasing role in warfare in the previous two decades. Most of these early approaches, especially those UAVs designed for operational mission purposes, focused on heavily equipped UAVs engineered to support a wide range of missions [1]. However, the downside of this multi-mission capability is the increased cost in development, test and evaluation, and acquisition, as well as increased demand for personnel to operate them. As a result, only industrialized countries have previously been able to afford UAVs and/or have the technical knowhow to build them.

Nowadays, in contrast, inexpensive and easy-to-produce UAVs are readily available, either through commercial or public domain avenues, which makes them accessible and usable for many more groups and nations. Such availability of these emerging technologies inevitably will also change the existing doctrine or generate new doctrine. As the capability evolves, operators may not assign only a single UAV to a mission, but rather will use multiple assets for one mission in order to increase the probabilities for success. This trend towards increasing numbers of employed unmanned systems, together with the fact that UAVs, especially the smaller, tactical variants, are challenging to detect, especially smaller, tactical variants, poses a future threat to allied forces in military contexts. This emerging threat motivates the study of the strengths and weaknesses of such use of unmanned systems, especially for analyzing their capabilities and potential opportunities to use them in defensive contexts as well.

A potentially interesting concept in this sense is the use of an autonomous network of such UAVs, where every UAV is an agent in the network and is able to communicate with all other UAVs. The level of autonomy is such that the network of UAVs can all work together and self-organize in order to accomplish a common mission, in which case we can describe this network as a UAV *swarm*. An analogy in nature is an ant colony where all ants follow one goal, for example, the survival of the colony. A mission where a UAV swarm could be appropriately applied may be the protection of a certain area against intruders, where each UAV searches in a separate part of this area and seeks to detect an intruder. If it detects one, then the UAV attacks and destroys the intruder. On the other hand, if there is more than one intruder, the defending

UAV can call for help and tracks the intruders until all other UAVs gather and prepare to attack the intruders all at the same time. A reasonable question in this hypothetical scenario is whether this latter tactic of amassing forces is more efficient than attacking an intruder immediately when one is detected.

Such questions of tactical approaches are increasingly important and merit study. In particular, tactical approaches become more challenging in the case where the intruders are also numerous and are as capable as allied agents. We can then call this a *swarm versus swarm* scenario, which is the subject of deeper study in this thesis.

1.2 Objectives

To date, air battles have only take place between manned fighters. However, with the increasing number of UAVs in the battle space and the increasing number of countries which use UAVs, the chances for combat engagements between UAVs are also increasing. Some relevant questions include: what are the factors that makes one UAV a winning UAV, and what are the differences between traditional air battles and future ones involving UAVs? Will the tactics be different? If an autonomous swarm of UAVs is involved, employment of *swarm tactics* could make the difference in these cases.

One way to explore the area of UAV swarm tactics is by using simulation, especially if there is a capability gap in reliable experimentation of UAVs capable of interacting intelligently with other UAVs. A good choice would be an agent-based model (ABM) because agents in the swarm normally have a small set of rules and are guided by their local objectives. Note that this abstraction of a relatively simple agent supports the idea of easy-to-build, simple, and cheap UAVs, which can be used in a fire-and-forget sense. Another way to describe swarm UAV combat is by modeling it as a Markov process, where the states represent certain battle situations. In order to conduct this analysis, we require information regarding the transition rates at which state changes occur. However, such information is not currently available because of the scarcity of research in swarm unmanned system tactics. One idea is to estimate these rates by analyzing the ABM. Another way would be to look at previous research papers on manned air combat and adapt those results for our purposes. Feigin [2] and Nunn and Oberle [3] analyzed air combat engagements between manned fighter aircraft by using stochastic models. The advantage of the Markov process model is that it provides an analytic way to study such combat scenarios and represents them with a formal mathematical description.

With the combined simulation-based and stochastic model-based approach, rather than designing specific tactics which may not generalize beyond the investigated scenario setting, we use data farming methods, statistical design of experiments, and data analysis techniques to explore this area. In this manner, this work has the advantage of not solely relying on conventional air combat doctrine, such that the results can illuminate potential new insights into this new arena for swarm UAV combat.

1.3 Literature Review

The idea of swarms of unmanned vehicles is not new and has been extensively studied in a variety of contexts. Frelinger et al. [4] researched swarming connected to PRoliferated Autonomous WeapoNs, termed "PRAWNs." These weapons recognize their targets and fire at them. The implementation of cooperative behavior through communication between the weapons was shown to increase their effectiveness. A limited sensor range was compensated by communicating target detections to nearby weapons. The result was that more targets were hit in the target area and the system as a whole achieved better performance. We also implement this kind of behavior in the simulation models presented in this thesis, so as to explore how the disadvantage of less effective but cheaper and lighter sensors can be counterbalanced by communication.

Clough also describes this advantage of numbers and goes a little bit deeper to provide a reasonable working definition of a "swarm" [5], which we extend slightly in the next section for our purposes:

> A collection of autonomous individuals relying on local sensing and reactive behaviors interacting such that a global behavior emerges from the interactions.

This definition addresses issues of local awareness and reactive actions for individual agents, which contrasts with the need to maintain a global picture of the environment and conduct global predictive planning. In other words, we do not need every UAV to have the global view, but the swarm has a common picture of the battle space. This common picture enables the manifestation of what Clough and others calls an *emergent behavior*. In the presented approach, we also observe such a behavior upon implementing a model of communication between the UAVs. The self-organization (SO) of a swarm is closely connected to this. While every UAV can be seen as an individual, we want them to recognize each other and build a swarm of UAVs. Reynolds [6] describes this as flocking behavior which includes cohesion, alignment

and separation as the three main patterns of this behavior. Price [7] talks about this topic in great detail, using Reynolds' approach to UAVs as an example and providing a mathematical model to describe swarming. Besides different types of swarming and levels of SO, he also mentions kinds of communication to make necessary information available among the swarm members. A radio broadcast is one simple option that is also applied in this thesis. Another communication issue is the negotiation of target allocations, which Day [8] explored through his study and analysis of different decentralized and centralized algorithms.

Clough also mentions, among others, two suitable tasks for swarm UAVs which are of relevance to this thesis. The first one is Area Search and Attack, where a swarm searches an area for targets and attacks them after detection. The second is Surveillance and Suppression, where a swarm covers an area by possibly random search patterns and destroys every intruder. The random search pattern makes the swarm unpredictable in this case. These cases reflect both defensive and offensive tasks for swarms, and both are incorporated into the scenario studied in this thesis. Other works have studied analogous scenarios, such as in [4], where an air defense battalion in five clusters of ten elements in each cluster are attacked by PRAWNs. Beyond hypothetical scenarios and associated technologies, real-world operational contexts exist to motivate this study. For example, the *Harpy* UAV, developed by Israeli Aerospace Industries, is designed to attack enemy radar stations, and is nominally able to operate in large numbers in an offensive context [9]. This is one example of an existing threat posed by large numbers of weaponized UAVs working together in order to accomplish one common mission. The described threat motivates the operational scenario we use in this thesis.

In recent news, we see the first examples of manned fighters in engagements with unmanned drones [10, 11]. However, these fights are limited to cases where the drone remains in a defensive position where the manned fighter tries to shoot it down. However, some views have been offered and some attempts made to enable UAVs for air-to-air combat missions [1], and [12] posits that our armed forces will see such capabilities within the next ten to twenty years. As such, this thesis addresses this future concept beyond the current capabilities of present-day technologies, which can further provide insights to guide development of and investment in for such future systems.

Despite the lack of substantial research in unmanned aerial fighter combat, we can learn from manned fighter combat tactics, for example, as described by Shaw [13]. Shaw discusses both dogfighting maneuvers and division tactics. "Dogfight" is a term that describes maneuvers

in a one-on-one fight, whereas "division tactics" concerns the behavior of a group of fighters working together as a unit. The models of combat we use in this thesis include a limited implementation of dogfighting behavior, but it is not the primary focus of this research. However, we should keep a deeper exploration of basic fighter maneuvers from the manned fighter world in mind as a potential future analysis question, in which a sensitivity analysis could be done to see if different employment of basic maneuvers changes the results dramatically. In contrast, given the interest of this thesis in investigating swarm tactics, this study specifically implements elements of division tactics.

To date, there are no real-world operational examples of swarm UAV combat available. However, swarming in warfare is not restricted to robotic forces. History provides numerous battles where at least one force can be considered a swarm, as studied extensively by Edwards [14]. He explains that "a swarming case is any historical example in which the scheme of maneuver involves the convergent attack of five (or more) semiautonomous (or autonomous) units on a target force in some particular place" [14]. With this definition, Edwards identified ten historical battles from ancient times to the recent past. The most interesting ones are the horse-archer cases in the Eurasian steppe. Light but fast horsemen armed with bow-and-arrows surrounded the enemy force and converged, shot, and departed continuously by staying outside of the enemy's weapon range and using their own superior range. Edwards identified three factors - elusiveness, longer range of firepower, and superior situational awareness - as key factors enabling the swarm's success. The work done by Day [8] also demonstrates the importance of these key factors for UAV swarm versus UAV swarm battles, where Day identifis speed, which relates to elusiveness, and probability of hit, which is connected to longer range of firepower, as key factors for swarm UAVs. However, in the case of this thesis, we assume that both sides are equipped with the same type of UAV with no platform-based advantages on either side. Therefore, we expect that all three factors do not come into play. Rather, we investigate the described tactic, such that we anticipate that this should give an advantage at least to those UAVs who attack from the rear in the first phase of the battle. Edwards also makes another differentiation between dispersed swarm and massed swarm approaches [14], where dispersed swarms are more appropriate for modern combat because of the effect of weapons of mass destruction that can inflict wide-area damage and casualties. As we do will not have such wide-area weapons modeled in the presented scenario, we neither expect show this absence through simulation analysis, nor do we expect to see a difference in our case between dispersed and massed swarm tactics.

Further, Sanchez and Lucas define agents as entities which "are aware of (and interact with) their local environment through simple internal rules for decision-making, movement, and action" [15]. This complements the definition of the UAVs we are thinking about and the modeling approach for this thesis. Simple and inexpensive UAVs operate in an autonomous and self-organized manner to accomplish the mission. Hence, agent-based simulation (ABS) seems to be a reasonable choice of tool. There are plenty of ABSs available, including some reviewed by Nowak [16] with respect to modeling capabilities for swarm UAVs. He mentions MASON (Multi-Agent Simulator Of Networks) as a good framework for this purpose, though it is not without some gaps in analytical capabilities, visualization, learning algorithms, visualization and kinematics. Upon review, these limitations in earlier versions of MASON have largely been addressed through further development of the software. MASON is designed to maximize execution speed [17] and comprises a 3D representation of the environment. These are the main reasons for the decision to use MASON for this research.

1.4 Scope of the Thesis

The area of swarm UAVs offers a wide range of research opportunities which have been rarely explored so far. We focus on the tactics of a UAV swarm in this paper. The proposed scenario envisages two phases, one which requires search and detection of the adversary's UAVs and the second which specifically investigates the interaction between opposing UAVs. This work focuses on the latter air-to-air engagement, and employs simplified search and detection models for the former.

Further, there exist many diverse UAV platform types which differ in size, weight, body structure, aerodynamic behavior, and so on. Another issue is that current UAVs rarely possess air-to-air fighting capabilities, such that we are analyzing a future system where UAVs are endowed with such capabilities. As the focus of this research is on tactics and not platform design, we concentrate on a model of a representative UAV with given and fixed platform characteristics to be detailed later.

1.5 Course of Study and Methodology

The importance of UAVs is increasing in military missions and significant effort has been invested to raise the level of autonomy of UAVs. One goal is to develop fully autonomous swarms. If we get to such a level, then one possible scenario is the proposed swarm versus swarm combat setting. We want to address the question what such an engagement would look like. However,

we note that there is almost no data available because current UAVs lack such capabilities. Simulation is one reasonable way to obtain reasonable insights given the absence of real-world data. One also obtains some benefits from simulation that real world experiments do not offer. With simulation, for example, we have the opportunity to conduct many experiments, and we are not (as severely) restricted by weather, time, or money. For these reasons, one can explore innovative ideas beyond those of common knowledge and understanding to potentially provide more insights. The drawback is that simulations or models (more generally) are abstractions of the real world with more or less reasonable assumptions, which have a significant impact on the results. Therefore, these assumptions have to be chosen very carefully and should be documented, as is done in this thesis.

The first step in the presented approach is to define an operational scenario for our purpose. Let us consider two high value targets of opposing forces, a Red and a Blue one. Both have the ability to launch a swarm of 50 UAVs, which are able to cooperate among their respective agents. The mission is to protect their own home base (i.e., the high value target) and to destroy the opposing one. Each side has the same type of UAV with the same specifications. The scenario starts with UAVs already launched. During the experiments, we vary relevant factors such as the initial positioning, spatial and temporal coordination, number of flights, and tactical behavior.

In order to model the scenario we use the discrete-event multiagent simulation library, MASON. The advantages of MASON are the fast execution times, the provided 3D environment, and the separated visualization from the model implementation itself. The agents are the key part of the simulation. They implement the main logic which describes the scenario, but in the end are guided by simple rules like search, destroy, stay close to your neighbors, etc. For that purpose, we implement the agents (UAVs) by writing sub-models for motion, cohesion, communication, search and detection, target assignment, combat and tactics. Initially, we do not know to what degree of fidelity the sub-models must be to provide realistic representation while garnering useful insights. Therefore, we construct the sub models in an incremental, iterative manner, which means we implement the sub models with very simple functionality in the first step and progressively enrich the sub models. For example, the initial search and detection model uses a cookie cutter sensor to model perfect, yet perhaps unrealistic, sensing. After the analysis of the simulation output, we assess whether we implemented a model which allows us to answer the research questions or we have to refine the sub models. The simulation itself should be designed with this modularity in mind, and to facilitate both easy input and output, should be able to read

comma-separated values (CSV) files with sets of initial values, to run the model with every set several times, and to write a CSV file with the results.

Rather than scripting the agent behavior deterministically, the agents should be endowed with sufficient logic to accomplish the mission on their own only depending on a set of initial values. Each possible set of initial values is one setup for the swarm. If we screen all sets, then we can identify important factors which are crucial for mission success within the simulation and for answering the proposed research questions. Unfortunately, given the numbers of factors and the magnitude of all possible permutations, we are not able to run the simulation with every possible set to do this in a feasible amount of time. Therefore, we have to choose carefully through the use of design of experiment techniques, which allow the analysis of the whole factor space by using a much smaller number of design points. Latin hypercubes are one method that provides the opportunity to scan the whole factor space efficiently. Given initial insights into swarm behavior from this study, additional richer methods may be applied for exploring a subset of the factor space more deeply.

After the simulation runs, we analyze the output data, and in order to do so, we need to identify appropriate response variables, as defined by measures of effectiveness (MOEs) and measures of performance (MOPs). We are mainly interested in the outcome of the swarm versus swarm engagement; therefore, the first measure is who won the battle. The response variable would be either zero for a Red success or one for a Blue success. This is a binary variable, and we can use logistic regression. Another metric is the number of remaining UAVs on each side. Linear regression should be applied in this case. Based on the statistical analysis, we are able to determine which sets of factor values are the most promising ones and derive insights from them.

1.6 Organization of Thesis

In Chapter 2, we provide a description of the scenario that was used for our research. Then we develop the simulation model. We start on the application level, describe the the integration of the model in the used framework and explaining some basic concepts we are using. Later we give the mathematical descriptions of the programed modules which provide the agent behavior. In the third section of Chapter 4 we talk about the Markov process. We use a bottom-up approach in this case. First one-on-one combat is modeled and the results are integrated in a multi-agent engagement.

The experiment we want to conduct is described in Chapter 3. There we explain the independent and dependent variables. Furthermore, we introduce the two designs that were run on the simulation system.

Chapter 4 is divided into two parts. The first leads us through the analysis of the output of the simulation model. We look at the raw data as a first step. Then we solve some issues in the regressors, followed by regression it self. We finish the section with an exploration of additional simulation runs where we apply findings from the former analysis. The second part analyzes the steady states of the analytical model and provides sensitivity analysis of transition probabilities.

We connect the findings in Chapter 4 and gives recommendations for tactics in swarm versus swarm combat. We also identify issues with the current models and give ideas for further research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Model Formulation

An expanded description of the presented swarm-versus-swarm scenario is contained in this chapter, followed by a detailed outline of the simulation and analytical models explored in this thesis. The simulation model provides the flexibility to investigate the complex scenario while doing so in a methodical manner through design of simulation experiments. We also examine whether an analytic model based on a stochastic process representation can provide similar or alternate insights..

More specifically, we analyze swarm tactics using a 3D agent-based simulation model based on the simulation software framework, MASON [18]. In this simulation, the UAVs are defined as agents, where sub-models for motion, sensors and combat define the individual agent behaviors. The swarm itself is based on the emergent flocking behavior and assumed communication between the agents.

The second modeling approach is a stochastic process model based on Markov transitions between discrete states. We define phases for UAVs in swarm engagements, and including non-fighting UAVs (i.e., those in transit) and those already downed (e.g., by the opposing forces). We assume that engaging UAVs meet with a given probability and commence the swarm combat sequence. As these swarm engagements can often be decomposed into one-on-one dogfights, we can define states such as when one of the two UAVs defeats the other and proceeds to seek another opposing UAV to engage. The steady state for this model gives the probabilities for Red and Blue forces to win the battle.

2.1 Scenario and Related Motivation

The main purpose of this thesis is to analyze a UAV swarm as a countermeasure against threats imposed by a hostile UAV swarm. The research community has worked on autonomous UAV swarm capabilities for some time, including research into swarms for air-to-ground missions. However, very few works have investigated air-to-air engagements for unmanned combat aerial vehicles, let alone swarms of UAVs. In practice, UAVs capable of fighting against other unmanned aerial vehicles are also unavailable at this time. However, given the motivating trends in unmanned systems in warfare, UAV swarms in general and, in particular, their use in air-to-air combat are envisioned to be part of future military missions.

To analyze such missions, we define a scenario with a Blue and a Red swarm, each consisting of 50 UAVs, with each UAV equipped with weapons, and their respective home bases, which represent high value targets (HVTs). The UAVs act as autonomous entities of a swarm with capabilities for communicating with one another and correctly recognizing friendly and opposing UAVs. The area of operations is designed to be ten by ten kilometers, and both bases are placed on opposite sides of the area, ten kilometers apart. The air space is designed to be four kilometers high. Both swarms are assumed to have already been launched and are initially loitering in the space above of their home bases at the beginning of the scenario. It is assumed that both sides have perfect knowledge, presumably through reconnaissance, of the position of the enemy home base. Figure 2.1 gives an idea of what the situation looks like at the beginning of a simulation run. Blue and Red start their attack at the same time.

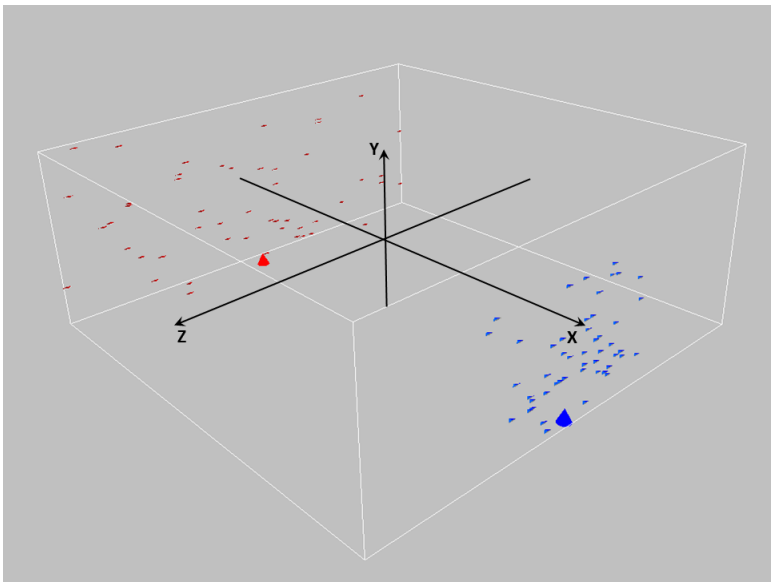


Figure 2.1: Illustration of the envisioned battlespace environment for the swarm-versus-swarm combat scenario, including definition of the coordinate directions.

The main objectives of each swarm are two-fold: to destroy the opposing home base through a "suicide" UAV attack and to protect their own home base by engaging inbound enemy UAVs and shooting down them. Note that victory condition is defined as when the enemy's HVT is destroyed (i.e., hit beyond the HVT's staying power) or if the enemy swarm is destroyed (i.e., no adversary UAVs remain).

The relevant MOEs for this scenario include the probability that blue wins the battle and the expected numbers of downed blue and red UAVs, respectively.

2.1.1 Statement of Assumptions

The described reference scenario represents a simplified environment, which does not include any external features like different terrain or other kinds of obstacles. It is also assumed that both swarms consist of the same type of fixed-wing UAV with exactly the same platform capabilities, and are further assumed to be basic platforms that are easy and inexpensive to produce, thereby enabling expendable employment. In addition, both home bases are assumed to have no other defensive or offensive capabilities beyond their respective swarm of UAVs.

In order to keep the computational effort as small as possible, we assume perfect conditions for communication, reconnaissance, and shooting. It is possible to scale the radio, sensor, and weapon ranges as necessary, but we do not implement any distribution that models the loss of precision with greater distance. In this manner, all three components can be seen as cookie cutter models.

Further, launching and recovery of UAVs is not modeled; instead, it is assumed that all UAVs have already been launched at the beginning of the scenario and fly and fight until they either get shot down or one side has reached the main objective. Fuel (endurance) and ammunition consumption (weapons availability) are not included in the model. Also, both swarms are initially positioned within a defined space near their respective home bases.

A pre-launched swarm seems unrealistic as this would reduce the endurance when an attack is really conducted. On the other hand a dispersed positioning is time consuming when the UAVs get lunched right before the mission starts. This would limit the options to just a central setup. One way to get around these issues in real world are several launchers which launch UAVs parallel. This saves time and the launchers can be prepositioned itself to support the preferred positioning of the UAVs.

2.2 Simulation Model

Several simulation design questions arise when developing such simulation models. We state that the overall goal is to provide a simulation environment that gives the ability to analyze different swarm tactics based on various parameters of interest. Therefore, we require data farming capabilities, in which case the computational runtime of the simulations is a significant concern. We will always find a tradeoff between realism and abstraction in this sense. The primary questions include: How much realism is needed, and is it worth the increase in model complexity? What are the limits of the insights obtained from different levels of fidelity of the

simulation model?

Another important aspect of simulation design is the construction of the environment itself. Should we use a two- or three-dimensional space? Given the fact that UAVs operate in a three-dimensional space naturally and the positioning of units is integral in the definition of tactics, there is no reason to believe that analysis in two dimensions will provide the same insights as in three. Therefore, we consider a 3D environment in which to model the swarm versus swarm engagement scenario.

In the previously stated assumptions, we note that the swarms on both sides comprise identical, inexpensive, and easy-to-produce UAVs. These UAVs are endowed with simple behaviors defined by a rule set, for example, including:

- Stay close to your teammates,
- Avoid collision among teammates,
- Attack an opposing target,
- Help your teammates,
- Detect opponent UAVs

Even if the rules are basic, the interactions between the UAVs can be complex, as mentioned by Sanchez and Lucas in their paper [15]. It is then an obvious choice to use an agent-based model (ABM) to consider each UAV as an individual agent.

2.2.1 Agent-Based Simulation

Agent-based simulation (ABS) can be considered a bottom-up simulation modeling approach. Objects of the real world are modeled as agents. These objects are mostly individuals who are able to think, make decisions, and interact with the environment or other objects. Some common real-world examples include a flock of birds, a colony of ants, or a crowd of humans. An agent is modeled as being driven by its local preferences, which can potentially be numerous and not necessarily of all equal weight. The agent observes the environment with sensors, and is also assumed able to communicate with other agents to get relevant information that affects its behavior. All of these influences are analyzed in a decision process and are used to determine the next action for the agent to perform. This action, in turn, further influences the environment, and iteratively impacts the dynamic scenario, including its own future actions. A visual block diagram explanation of this iterative dynamic process is shown Figure 2.2.

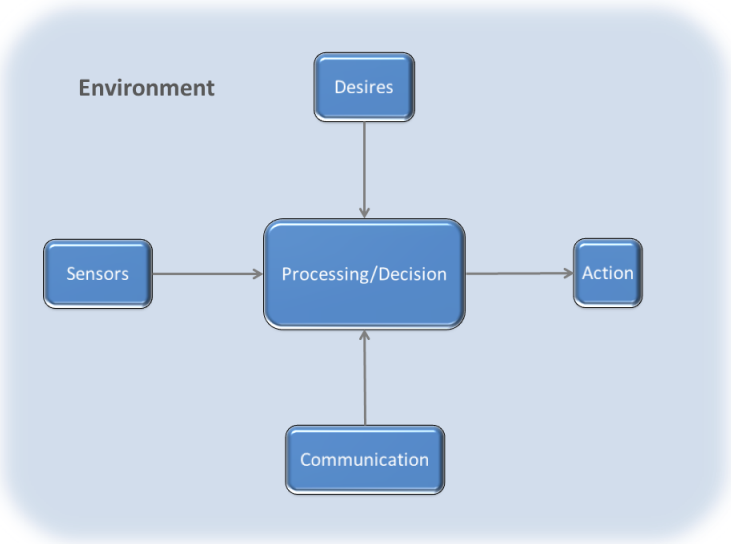


Figure 2.2: Information flow for an individual agent in an agent-based simulation model.

Despite the simplicity of the proposed behavior rule sets, the decentralized implementation and further implications for the simulation can be quite complex, as discussed in, for example, Cioppa, Lucas and Sanchez [19]. ABS is often the first choice in problems connected to swarm behavior. We want to take advantage of these simple rules and their employment on simple, cheap and easy-to-produce UAVs which could benefit from swarming. As a matter of fact, we can observe such behaviors in natural systems as well, such as with a school of sardines in the ocean. They are nominally simplistic creatures with limited sets of behaviors, including an instinctual behavior to stay close together and to evade predators. Their sensors are their eyes with which they recognize other fish and react according to their local preferences. The resulting school collectively moves in complex shapes that we would not expect at first glance.

2.2.2 Simulation Software Environment

Fast runtime execution, representation of a 3D environment, and the ability to implement agent-based behaviors are the three characteristics which are important for the simulation environment. MASON (Multi-Agent Simulator Of Networks) is a discrete-event simulation framework that satisfies all three. According to Luke et al., MASON is designed to meet the computational demands for multi-agent systems [20]. It also provides a 3D environment with modular visualization (i.e. the visualization engine is separate from the agent behavior logic). The framework is an open-source Java project out of George Mason University with a minimal model library. Experienced programmers can easily add features and adapt the environment to their custom

needs. Our own experience with MASON shows that the framework provides exactly the right amount of functionality to setup a basic model immediately. It is well documented and provides instructive examples, which are very useful to understand the concepts.

Software Architecture

The simulation designed in this thesis is developed using Eclipse [21], a commonly used environment with many development tools for Java projects. MASON is essentially a Java library and can be imported into Eclipse. The Java 3D application programming interface (API) is needed for the 3D support and the Java Media API enables multimedia capabilities such as movies, charts and graphs. A brief installation how-to for MASON, discussion of additional APIs, and a tutorial of a simple existing simulation model are provided in Appendix A.

The simulation software consists of four classes that define the swarm engagement scenario of interest to this thesis:

1. `HVT.java`: This class implements the agent that represents the home bases. It is more or less a dummy agent because the bases have no functionality at all and represent the target for the UAVs. Without visualization we would not need this agent type. However, future extensions for mobile targets, for example, can benefit by instantiating agents of this object class.
2. `UAV.java`: This class contains all the logic for the individual UAV and is therefore also an agent type. The UAV agent is essentially the core of the simulation model as this class implements all the functionality that makes a swarm of UAVs out of a collection of agents. Given its importance, we discuss this class in more detail in the next section.
3. `UAVSwarm.java`: This class defines the battle arena and controls the simulation flow, and thus provides the entry point to inclusion of variable parameters in the simulation environment. The `main` method contained in this class starts the simulation and generates an instance of `UAVSwarm` by calling the `doLoop` method with input argument `args`, which controls the simulation:

```
635     args = new String[]{
636             "-repeat", String.valueOf(1000),     //how often each row of the CSV file
637             "-time", String.valueOf(0),        //no messages during the run
638             "-until", String.valueOf(3000),      //max time for each run
639             "-seed", String.valueOf(1366691235073L) //seed for the first replication
640           };
```

   The parameters `repeat` defines the number of simulation runs, `seed` specifies the seed, `until` stops the simulation run after a certain simulation time, and `time` can be used to

provide information about the state of the simulation in periodic intervals. These parameters can be used to assist in the experimental design for simulation studies.

4. `UAVGUI.java`: This helper class comprises necessary code to visualize the simulation run (see Figure 2.5). It initializes an instance of `UAVSwarm` and runs it. Since visualization is not the main objective of this thesis, we do not explain this class in detail. However, the interested reader can find all code in Appendix A.

3D UAV representation

The battle space is a 3D environment; therefore, we model the UAV as a three-dimensional rigid body. A simple way to do this would be to model each agent as a sphere with an attached arrow that describes the heading of the UAV. However, for the sensing and combat models, we wish to address the variable projected surface area (e.g., radar cross section) for different aspect angles. In other words, the probability of hit by an attacking UAV's weapon is assumed to be influenced by the exposed surface area of the attacked UAV.

Even though no current UAVs have such swarm combat capabilities, we present a simplified geometric UAV design, as shown in Figure 2.3, as a platform shape similar to those currently in development for live-fly experiments of this future concept of swarm-versus-swarm UAV combat (see Figure 2.4).

The illustrated flying wing aircraft has a wing spread of 10 meters. Later, in Section 2.2.3, we use the area shown in the three two-dimensional plots to determine the probability of hit. Here, the area as seen directly from above or below (Figure 2.3[a]) is $A_1 = 18.75$ m, from the nose or tail (Figure 2.3[b]) is $A_2 = 7$ m, and from left or right sides ((Figure 2.3[c]) is $A_3 = 3.56$ m.

Initial Positioning of Swarm UAVs

Section 2.1.1 states that the UAVs are already launched and remaining near their respective home bases at the start of the simulation. The initial positioning of the elements of the swarm represents a *tactical* decision to spatially amass or disperse forces, and thus, is a specific factor of interest which we vary to observe its impact on overall mission effectiveness.

In reference to the simulation software, the arena is defined within the `UAVSwarm.java` class, in which the `start` method (see the illustrative example below) first initializes the dimensions of the area of operations.

```
public void start() {
  Continuous3D yard = new Continuous3D(1,10000,10000,4000);
```

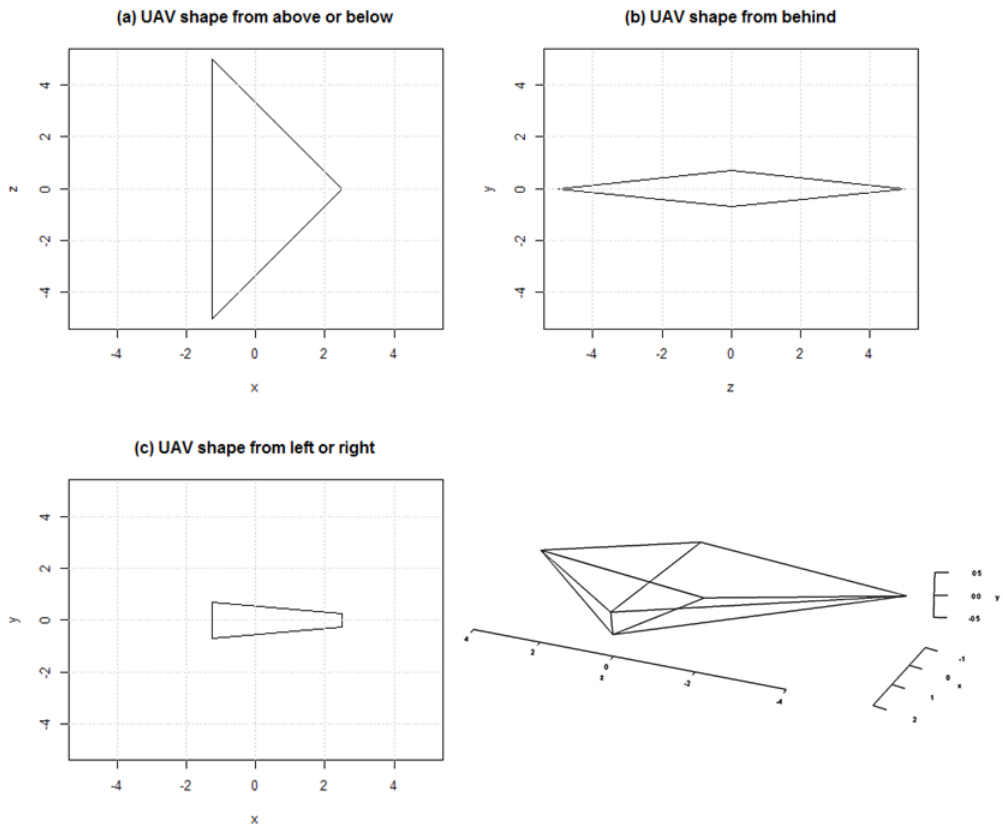
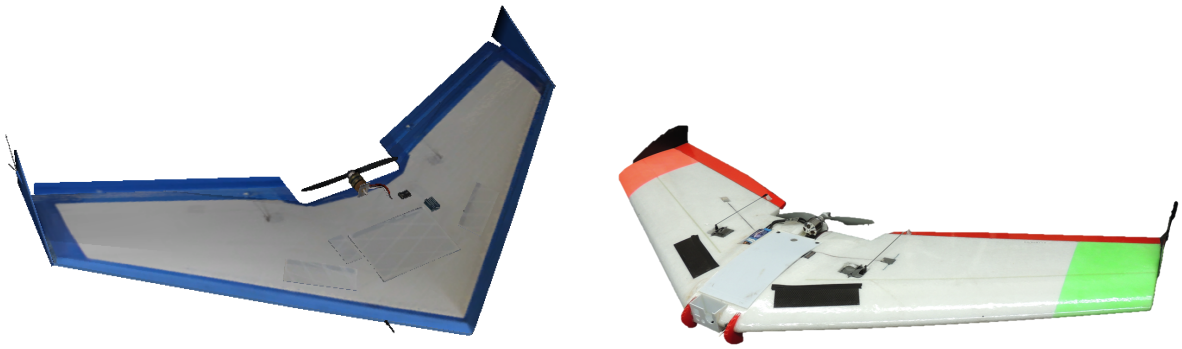


Figure 2.3: Shape of the simplified simulated UAV platform, projected in two and three dimensions. Different approach angles yield different projected cross sectional areas, which influence factors such as probability of successful hit by enemy fire.

```
    UAV uav = new UAV(27,0,0);
    yard.setObjectLocation(uav, new Double3D(0,0,0));
    schedule.scheduleOnce(uav,0);
}
```

The first line identifies an instance of `Continuous3D` that defines a field of 10 by 10 by 4 kilometers (where the first argument is the discretization length, a value of one meter resolution in this case). The origin is placed in the center of the field, such that the boundaries of the arena are defined as $x = \pm 5000$ meters, $y = \pm 2000$ meters, and $z = \pm 5000$ meters in a right-handed coordinate system.

The remaining lines in the above `start()` method allows initialization of UAVs within this arena (in this example case, only a single `UAV`) and place them at a specified position with the `setObjectLocation` method (in this case, at $(0,0,0)$). As a note on agent-based simulation



(a) (b)

Figure 2.4: Images of two current research UAV platforms, namely the (a) Procerus *Unicorn* UAV and (b) modified Ritewing *ZephyrII* RC plane, for ongoing live-fly field experimentation of the swarm-versus-swarm combat scenario.

execution, the method `scheduleOnce` puts the agent in the event queue to execute this agent's behaviors in each simulation step.

The battle arena for the given dimensions and coordinate frame, as well as the initial configurations for both UAV swarms and high value units are shown in Figure 2.5. The home bases are positioned near the minimum and maximum limits along the x-axis and laterally centered.

Figure 2.5 also annotates the regions within which the UAVs can initially be positioned, such that randomized variations in starting locations can be studied as a matter of tactical relevance. The starting bins are defined by the volume comprising the whole width along the z-axis, the whole height along the y-axis and 1000 meters in depth from either limit along the x-axis.

Due to the assumed symmetry in the capabilities of both Red and Blue swarms, we can investigate variations only on one side (Blue) while initially assuming a nominal deployment by the other (Red). In this manner, assume that the Red UAVs are initially positioned in their starting region uniformly randomly, where $p_{P_{initialRed}} = \left(x_{P_p}, y_{P_p}, z_{P_p}\right)^T$ and

$$
\begin{aligned}
x_{P_p} &\sim U\left(-5000, -4000\right) \\
y_{P_p} &\sim U\left(-2000, -2000\right) \\
z_{P_p} &\sim U\left(-5000, 5000\right)
\end{aligned}
$$

This scatters the Red agents in the swarm uniformly within the bin for every simulation run.

In contrast, we provide additional degrees of freedom for the initial positioning of the Blue

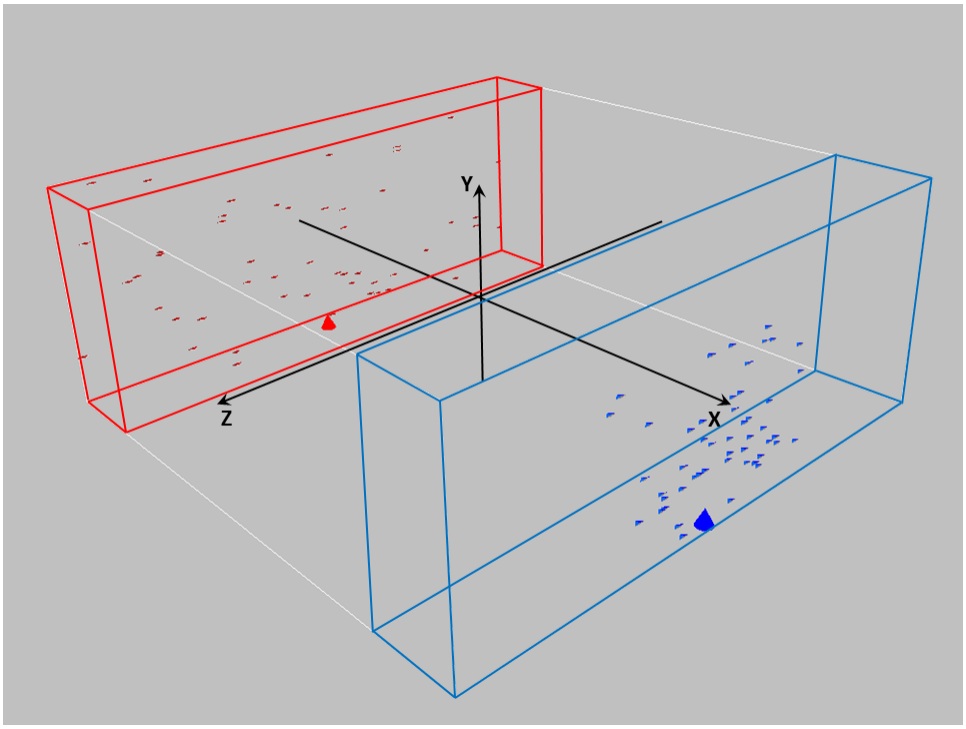


Figure 2.5: Battle arena with Blue and Red home base and 50 UAVs on each side. The Blue and Red box show the space for the initial positioning of the Blue and Red swarm, respectively. The used coordinate system is also mentioned in this figure.

UAVs, by using the beta distribution to (independently) randomize the starting y- and z-coordinates. The x-coordinate distribution for the Blue UAVs, like their Red counterparts, is distributed uniformly. Thus, we have that

$$p_{P_{initialBlue}} = \begin{pmatrix} U(4000, 5000) \\ Beta_{-2000,2000}(\alpha_y, \beta_y) \\ Beta_{-5000,5000}(\alpha_z, \beta_z) \end{pmatrix}$$

A beta distribution has nice properties which we will use for the design of experiment and the analysis. The goal is to explore various initial positions of the Blue swarm against a Red swarm whose collective initial positions are always uniformly distributed. Rather than being restricted by a handful manually selected positions, the whole space, at least in y- and z-directions, should be explored, and the beta distribution provides a parametric ability to perform such a diverse study.

The choice of the beta distribution merits some additional discussion. It takes two shape pa-

rameters, $\alpha > 0$ and $\beta > 0$. Their ratio and magnitudes define the shape of the distribution. Some examples of how different values for α and β influence the silhouette of the probability density function are shown in Figure 2.6. The random variable $X \sim Beta(\alpha, \beta)$ is defined on the interval $[0, 1]$ and can be scaled to any dimension needed.

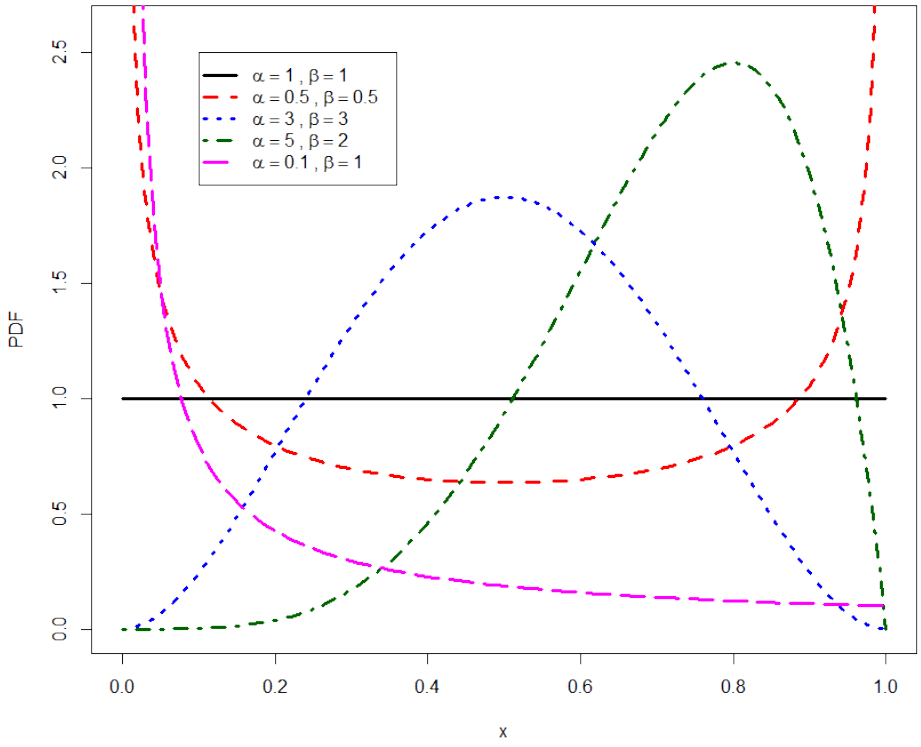


Figure 2.6: Beta distribution for different values of α and β.

We can explore how different shape parameter combinations can be related to various operational implications. For the purposes of the following discussion, we concentrate on the horizontal z-axis, since this dimension likely has a larger tactical impact due to its larger spatial extent (compared to the y-axis).

- $\alpha = \beta = 1$: As seen in Figure 2.6 (black line), these values cause the Beta to reduce to a uniform distribution. The swarm is equally distributed over the whole width of the area of operation. Therefore, all UAVs start their attack on the opposing HVT at the same time, and there is no concentration of forces in this case.
- $\alpha = \beta > 1$: With the resulting shape of the initial positions closer to a unimodal distribu-

tion, the UAVs are more clustered about the home base. The amassed forces are therefore in the center and represent a more direct route (i.e., shorter distances) for the UAVs to follow to the adversary's home base.

- $\alpha < \beta$ or $\alpha > \beta$: These values create a positively or negatively skewed distribution, which puts the main effort on the left or right flank, respectively. The main force conducts a flank attack with maybe less resistance but longer distances to transit to the adversary's home base. This tactic could give an opponent that is attacking more directly down the center a time advantage. However, depending on the magnitude of the shape parameters, there may be more or fewer defending UAVs remaining in the center and the alternate flank. These UAVs could be interpreted as a screening force that delays attacking enemies.
- $\alpha = \beta < 1$: In this case, the forces are concentrated on both flanks (instead of retaining forces in the central area). This initial position might be advantageous for an adversary predominantly in a defensive posture with its forces amassed centrally (i.e., not expecting attacks from the flanks).

The stated interpretation for parameter value combinations can be easily translated to the vertical axis (y-axis), where a beta distribution is also used.

2.2.3 Modeling of Individual Agents

We next describe the behavior models for individual agents, the collection of which forms the swarm of UAVs. The individual agent is endowed with the ability to move, to recognize its neighbors, to react to changes in the environment, and to coordinate its actions with some or all of its swarm teammates.

In this section, we describe the formal specification of the agent model which we implement in `UAV.java`. These mathematical models are implemented in sub models according to the description below. As a software implementation note, the `step()` method within each instance of the `UAV` object is repeatedly executed by the simulation at each simulation time step in a event-based manner in order to run the behaviors of each agent, thereby executing all sub models contained within.

The breadth of the parameters that can be adjusted for each UAV object within the simulation summarized in Table 2.1. The notation for the various parameters will be used in the descriptions and analysis throughout the remainder of this thesis.

Parameter Name	Type	Description	Units
p_E	Position vector	Position of the opposing home base	$h_E^T = (x_{p_E}, y_{p_E}, z_{p_E})$
p_H	Position vector	The point in the space that the UAV tries to reach currently.	$p_H^T = (x_{p_H}, y_{p_H}, z_{p_H})$
p_P	Position vector	The position of the UAV at the current simulation step	$p_P^T = (x_{p_P}, y_{p_P}, z_{p_P})$
$\vec{V}$	Directional vector	Velocity vector	$\vec{V}^T = (x_{\vec{V}}, y_{\vec{V}}, z_{\vec{V}})$
v_T	Constant	Cruising speed	$meters/second$
v_{Min}	Constant	Minimal speed	$meters/second$
v_{Max}	Constant	Maximum speed	$meters/second$
v_{Up}	Constant	Minimum speed a UAV can accelerate to by climbing	$meters/second$
v_{Down}	Constant	Maximum speed a UAV can accelerate to by going down	$meters/second$
a	Constant	Acceleration factor	$meters/second^2$
a_A	Constant	Acceleration factor for alignment	$meters/second^2$
a_D	Constant	Acceleration factor for the case where the UAV goes down	$meters/second^2$
b	Constant	Slowdown factor	$meters/second^2$
b_A	Constant	Slowdown factor for alignment	$meters/second^2$
b_U	Constant	Slowdown factor for the case where the UAV goes up	$meters/second^2$
α_{Max}	Constant	Maximum angular velocity	$degrees/second$
s	Constant	Sensor range	$meter$
w	Constant	Weapon range	$meter$
k	Constant	Probability of kill	$meter$
d_{Min}	Constant	Minimum distance a UAV wants to maintain to all other UAVs in the swarm	$meters$

continued on next page

continued from previous page

Parameter Name	Type	Description	Units
d_{Max}	Decision Variable	Is the radius of a sphere around the UAV. The UAV wants to stay close to all UAV of its swarm within the sphere.	*meters*
d_c	Constant	Specifies a radius of a sphere around the center of mass. Where the center of mass is formed by all UAVs of the swarm within d_{Max}.	*meters*
d_H	Decision Variable	Distance in x direction at which the UAV starts to converge to the opposing home base (and therefore the swarm)	*meters*
n_A	Decision Variable	Maximum number of allocation per target	
f	Decision Variable	Weight factor between main objective and other targets	

Table 2.1: UAV parameter specification.

Motion Model for Bounded Curvature Flight

Given the assumption of fixed wing UAVs in the scenario, we note that a realistic model for the flight dynamics of the vehicle should be addressed, as the vehicle must nominally remain in motion to stay aloft. The flight dynamics of an airplane are dictated by the four forces: gravity, lift, thrust and drag. If we want to use this physical description, we would have to solve second-order differential equations, which would lead to a high computational effort, which opposes our stated goal of generating insights without suffering the computational complexity and runtime expense. Therefore we make the decision to use a simplified geometry-based kinematic model, which does not consider the above forces but provides enough detail to reflect turn times and spatial movement.

This kinematic description of fixed wing aircraft, also known as describing vehicles with bounded curvature, was first explained by Dubins [22] as part of a path planning problem in two-dimensional space and extended to three dimensions by Chitsaz et al. [23]. The vehicle described in the latter case is dubbed the Dubins' airplane. However, [23] does not go far enough for our purposes, which require additional extensions to address the dynamic environment of our swarm engagement scenario. Specifically, [23] assumed constant speed relative to the ground and decoupled changes in altitude as an independent degree of freedom. Rather, we combine all three dimensions and introduce the velocity vector of the UAV, denoted $\vec{V}$, that does not have to be constant throughout the UAV's trajectory. Then the augmented state Q of an UAV in a single simulation step t is described by the current position p_P and $\vec{V}$, that is,

$$Q = \begin{bmatrix} x_{p_P} \\ y_{p_P} \\ z_{p_P} \\ \vec{V} \end{bmatrix}$$

With this definition of the velocity vector, one can identify a heading angle, which defines a heading at each time step towards a specific point in space, denoted p_H. In general, given the objective to reach and attack the adversary's home base (located at p_E), the heading is nominally aligned in the direction of p_E. However, we note that, based on the current mode of the UAV as well as definition of the UAV behaviors, the desired heading could be towards other objectives. For example, for a given UAV agent, its p_H could be the center of mass (COM) of the friendly swarm, because the UAV has local preferences to stay close to its swarm teammates. p_H could also be the current position of an opposing UAV because the UAV has some incentives to destroy it. As a result, given the dynamically changing environment, p_H is never constant throughout the simulation run, and once p_H changes, the UAV changes its direction and turns toward the new p_H.

In the following, we explain how the augmented state, $\hat{Q}$, at the next simulation step, $t+1$, is computed. The directional vector $\vec{P}_H = p_H - p_p$ is defined as the vector which points from the current position to the desired position. In order to turn an angle α between the two vectors $\vec{V}$ and $\vec{P}_H$ has to be decreased by a certain amount. $\alpha_{\max}$ is the maximum angular velocity the UAV is able to turn within one simulation step, and it is, therefore, the maximum angle α can be decreased. $\vec{P}_H$ cannot be modified because the current position and the point of heading is

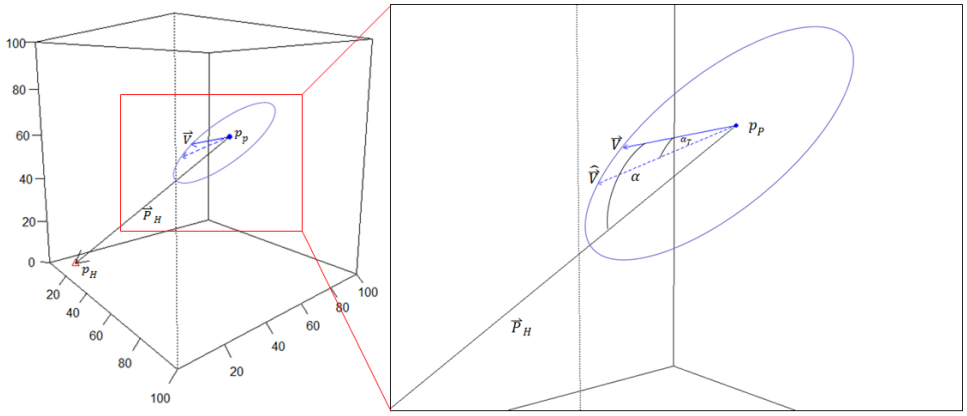


Figure 2.7: This is a rough picture of the idea of turning the velocity vector towards the vector defined by the current position and the point of heading.

fixed for this simulation step. But $\vec{V}$ can be updated in a way that α is decreased. The most challenging part is to figure out in which direction $\vec{V}$ should be turned, especially as we are in 3D space. At the end the new position of the UAV $\hat{p}_p$ is the point to which the updated velocity vector $\hat{\vec{V}}$ is pointing (see Figure 2.7).

In order to compute the new UAV position $\hat{p}_P$ and the new velocity vector $\hat{\vec{V}}$ we simplify the 3D problem into a 2D problem. $\vec{V}$ and $\vec{P}_H$ represent a plane in the space with normal vector $\vec{N} = \vec{V} \times \vec{P}_H$. If we consider $\vec{V}$ as y-axis of the plane's coordinate system given in 3D coordinates then $\vec{U} = \vec{V} \times \vec{N}$ is perpendicular to $\vec{V}$ and therefore the x-axis. Now we get the x-component of the 2D representation of $\vec{P}_H$ by projecting $\vec{P}_H$ onto $\vec{U}$ and taking the length of the resulting vector.

$$x_{\vec{P}_H} = \left| \frac{\vec{U} \cdot \vec{P}_H}{\left|\vec{U}\right|^2} \vec{U} \right|$$

The y-component is the length of the projection of $\vec{P}_H$ onto $\vec{V}$.

$$y_{\vec{P}_H} = \left| \frac{\vec{V} \cdot \vec{P}_H}{\left|\vec{V}\right|^2} \vec{V} \right|$$

$$\vec{P}_{H_{2D}} = \begin{pmatrix} x_{\vec{P}_H} \\ y_{\vec{P}_H} \end{pmatrix}$$

We also have to project $\vec{V}$ onto $\vec{U}$ and onto itself to get the 2D representation of $\vec{V}$. This is simpler because $\vec{V}$ is one of the coordinate axes.

$$\vec{V}_{2D} = \begin{pmatrix} 0 \\ \left|\vec{V}\right| \end{pmatrix}$$

We want to calculate the smaller angle between $\vec{V}_{2D}$ and $\vec{P}_{H_{2D}}$ and the turn direction. Both are given by the equation

$$\alpha = atan2\left(y_{\vec{P}_H}, x_{\vec{P}_H}\right) - atan2\left(x_{\vec{V}}, y_{\vec{V}}\right)$$

and the turn angle $\alpha_T = \text{sgn}(\alpha)\max\left(|\alpha|, \alpha_{\max}\right)$ is used to compute $\hat{p}_P$. In this case we use the parametric equation of a circle in 3D because the new position of the UAV is somewhere on the circle determined by p_P as center and $\left|\vec{V}\right|$ as radius. $\vec{V}_N$ and $\vec{U}_N$ are the normalized vectors of $\vec{V}$ and $\vec{U}$ respectively.

$$\hat{p}_P = p_P + \vec{V}_N \cos(\alpha_T)\left|\vec{V}\right| + \vec{U}_N \sin(\alpha_T)\left|\vec{V}\right|$$

The updated velocity vector results from the difference between the new and the old position.

$$\hat{\vec{V}} = \hat{p}_P - p_P$$

So far we do not have variability in the motion of a UAV as we see it in nature. Especially aircraft are affected by different air-streams, which makes it difficult to maintain a certain direction. We want to adjust the motion model to reflect this issue in the simulation. First we define an arbitrary unity vector $\vec{W}^T = (0,0,1)$ that could be seen as the wind direction before the current simulation step. The wind strength d_W is a uniform random variable between zero

and 10% of the current velocity.

$$D_W \sim U\left(0, \frac{\left|\vec{V}\right|}{10}\right)$$

To determine the new wind direction we use the spherical coordinate system where the origin is the current position p_p of the UAV. The angles ϕ and θ are again two uniform random variables.

$$\Theta \sim U(0, 2\pi)$$
$$\Phi \sim U(0, \pi)$$

They describe together with d_W the new position of the UAV. But before we determine $\hat{p}_p$, we update the wind direction.

$$\hat{\vec{W}} = \left|\vec{W}\right| \begin{pmatrix} \cos(\theta)\sin(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\varphi) \end{pmatrix}$$

Then the new position is given by $\hat{p}_p = p_p + d_W \hat{\vec{W}}$.

Modeling Climb and Descent

Another realistic issue is the impact of climbing or descending on the UAV's velocity (which is not addressed by Dubins' airplane models in [23]). We want to reflect at least a part of this effect of trading between kinetic and potential energy in our model.

We assume the UAV should not stop completely (i.e., as in a stall), and so introduce the minimum speed, v_{Up}. We also define v_{Down} as the maximum speed an UAV can reach as it descends. In the right-handed coordinate system, the y-axis determines the altitude of a given UAV and $y_{\vec{V}}$ shows the change in altitude in the current simulation step.

In the case $y_{\vec{V}} > 0$ the UAV climbs and is assumed to decelerate by a factor $b_U < 1$ only if the current velocity exceeds the product of v_{Up} and the ratio of the current velocity and its y-component.

$$\hat{\vec{V}} = \begin{cases} b_U \vec{V} & , \left|\vec{V}\right| > v_{Up} \left(\frac{\left|\vec{V}\right|}{y_{\vec{V}}} \right) \\ \vec{V} & , \left|\vec{V}\right| \leq v_{Up} \left(\frac{\left|\vec{V}\right|}{y_{\vec{V}}} \right) \end{cases}$$

In the worst case $y_{\vec{V}} = \left|\vec{V}\right|$ the UAV decelerates to v_{Up}. If the change in altitude is not so aggressive, then the UAV decelerates to a velocity between v_{Up} and $\vec{V}$. This takes into account that a higher speed can be maintained for a lower climb rate.

The same works for $y_{\vec{V}} < 0$ respectively, where $a_D > 1$ is the acceleration factor.

$$\hat{\vec{V}} = \begin{cases} a_D \vec{V} & , \left|\vec{V}\right| < v_{Down} \frac{\left|y_{\vec{V}}\right|}{\left|\vec{V}\right|} \\ \vec{V} & , \left|\vec{V}\right| \geq v_{Down} \frac{\left|y_{\vec{V}}\right|}{\left|\vec{V}\right|} \end{cases}$$

Parametrized Sensor Model

We assume that the UAV is equipped with a hemispherical cookie-cutter sensor with sensor range s. In other words, the UAV is able to detect every opponent UAV in its front hemisphere within s.

Algorithmically, the relationship between all UAVs in the space is initially stored in an undirected network, denoted Ψ, where every UAV has an edge to every opponent UAV. Based on this data structure, UAV i has now access to every hostile UAV j and can measure the distance s_j, for each i. If s_j is smaller than s for a given i, then an edge is drawn in a second directed network, denoted Γ, from node i to node j. This procedure assumes perfect identification of friend or foe. This method enables tracking of simulated detections amongst all UAVs of their opponents at each time step. Note that these detections are cleared at the beginning of every simulation step, and the procedure starts over. Further, we assume that all UAVs have instantaneous access to the collective set of detections taken by all swarm teammates because of the assumption of perfect communication among teammates.

2.2.4 UAV Swarm Modeling

Having adequately described the individual UAV agent models, we can begin to describe the swarm behaviors emerging from the collective local rule sets and the interactions among the individual agents. An assumption of the swarm is the ability for all UAVs to know their own positions and that of their teammates (or a subset thereof), made possible through the use of broadcast communications among the swarm. As will be described further in this section, each single UAV is now able to determine its relative position in the swarm and take actions to get closer, increase separation, and/or adapt speed and direction relative to its neighbors. Another advantage of this perfect communications assumption is the opportunity to allocate targets in an optimized way among the swarm elements, which we leverage in this work to present a decentralized solution.

Flocking Behaviors: Cohesion, Separation, Alignment

Craig Reynolds identified three key behavior patterns in his paper [6] for generating flocking behaviors: cohesion, separation and alignment. Their implementation is often termed "boids algorithm." In this section, we describe how we adapt Reynolds' idea to the proposed swarm UAV model by first describing these three relevant behaviors and their modified implementations in this work.

- Cohesion: Recall that the UAVs within a swarm possess a complete communication network to interact with each other. This network is represented by the two undirected networks Ω_B (for the Blue swarm) or Ω_R (for the Red swarm) where every UAV has a link to every friendly UAV. An arbitrary UAV at position p_{P_i} is aware of all positions p_{P_j} of all other UAVs j in the swarm. The sum of the positions divided by the number of teammate UAVs $n-1$, not including itself, gives the center of mass (COM), denoted c, for UAV i.

$$c = \frac{1}{n-1} \sum_{j=1; j \neq i}^{n} p_{P_j}$$

  We define d_c as the radius that specifies a sphere around c. One desire of a UAV is to stay in this sphere close by its swarm members. But c will certainly move towards p_E over time. Therefore the UAV makes a prediction and updates its desired heading target point, p_H, according to:

$$\hat{p}_H = \frac{1}{2}(p_H - c)$$

  Recall that the UAV has a cruising speed v_T, a minimum speed v_{Min} and a maximum

speed v_{Max}. In the case where the UAV is in front of c it decelerates by factor $b < 1$ until $\left|\vec{V}\right|$ reaches v_{Min}. If the UAV is behind the swarm then it accelerates by factor $a > 1$ until $\left|\vec{V}\right|$ reaches v_{Max}. The UAV slows down or accelerates by b and a, respectively, until $\left|\vec{V}\right|$ reaches v_T to achieve this cohesive behavior.

- Separation: The next step is to ensure that UAV i stays away a distance d_{Min} from its neighbors. First, each UAV identifies the closest UAV j and measures the distance, d_j. If d_j is smaller than d_{Min} then it turns $\vec{P}_H$ a certain angle β_T away from the vector $\vec{P}_{ij} = p_{P_j} - p_p$. This is done in the same way as described in the motion section above, such that the updated target heading point is given by:

$$\hat{p}_H = p_H + \vec{P}_{H_N} \cos(\beta_T) \left|\vec{P}_H\right| + \vec{U}_N \sin(\beta_T) \left|\vec{P}_H\right|$$

- Alignment: All UAVs follow one commonly known global goal, which is to target (and nominally destroy) the opposing home base located at position p_E. The UAVs always try to fly towards this point. Therefore, for the given swarm agents, we do not specifically need to address alignment behaviors because as long as the UAVs follow the global goal, they more or less line up automatically.

Target Allocation Optimization Model

In addition to motion control, we need a mechanism that assigns friendly UAVs to appropriate enemy UAV to engage and defeat, which is based on relative positions in terms of range to the target and relative bearing, γ_{ij}, as measured as the angle between $\vec{V}_i$ and $\vec{P}_{ij}$, the latter of which is the vector from agent i to enemy UAV j. The goal of the presented algorithm is to construct a decentralized approach to the allocation model. We note that the presented heuristic algorithm, iterated at every time step, provides a sub-optimal decentralized solution. A centralized method would leverage a network optimization algorithm that can provide optimal assignments; however, the decentralized architecture provides both robustness to a single point-of-failure as well as addresses the dynamically changing network topology in an efficient manner. For comparison of assignment performances, the centralized allocation is formulated and solved using numerical optimization software (i.e., General Algebraic Modeling System [**?**]). The results show that the objective value of the decentralized method is, on average, 20% above the optimal cost. This performance gap is deemed acceptable, given the advantages provided by the decentralized architecture, e.g., individual UAVs can optimize their allocations locally.

UAV i can use the appropriate network data structure, Ω_B or Ω_R, to identify all friendly UAVs

in its network, Γ, and therefore have access to positions of all detected enemy UAVs j currently in the collective field of view of the friendly swarm. Another directed network Λ keeps track of all currently allocated targets. Let m_j be the number of allocations for UAV j, and let $\left|\vec{P}_{ij}\right|$ be the distance between UAV i and j. Further, define f as the weight factor between the primary objective (i.e., targeting the adversary's home base) and other targets (i.e., targeting enemy UAVs). In other words, f represents the *preference* of the UAV to engage in an offensive role (f large) or a defensive role ($f \to 0$). Then the expression below

$$g_{ij} = \left(m_j + f\right)\left(\left|\vec{P}_{ij}\right| + \frac{\gamma_{ij}}{\alpha_{Max}}\left|\vec{V}_i\right|\right)$$

defines the desired objective function with a penalty for multiple allocations, m_j, for the same target. The main idea is that we want multiple allocations to ensure redundancy (i.e., high probability of kill for a given target) but not too many friendly UAVs allocated to one target to avoid the possibility of untargeted enemy UAVs.

In order to address the dynamically changing environment (since the enemy UAVs are also maneuvering in offensive and defensive roles), we also enable adaptive target allocations. To do so, define g_{Min}, which identifies the target currently with the lowest objective value, where $|p_E - p_P|$ represents the distance to the adversary's home base:

$$g_{Min_i} = \min\left(\left|p_E - p_{p_i}\right|, g_{i1}, g_{i2}, \ldots, g_{in}\right)$$

The UAV j corresponding to g_{Min_i} is considered a priority target, and an edge is drawn in Λ from i to j. However, given the operational constraint that limits the number of allocations per target, define this maximum number of friendly UAVs to target a single enemy UAV as n_A. In the special case where $f\left(\left|\vec{P}_{ij}\right| + \frac{\gamma_{ij}}{\alpha_{Max}}\vec{V}_i\right) \leq \left|p_E - p_p\right|$ and $m_j = n_A$, UAV i tries to find one friendly UAV l that allocates UAV j with $\left(\left|\vec{P}_{ij}\right| + \frac{\gamma_{ij}}{\alpha_{Max}}\vec{V}_i\right) < \left(\left|\vec{P}_{lj}\right| + \frac{\gamma_{lj}}{\alpha_{Max}}\vec{V}_l\right)$. Then UAV i deletes the allocation and allocates UAV j itself. In other words, target allocations can be traded among teammates if more (locally) favorable allocations are available. Finally, note that the target allocations for all UAVs are reset and recomputed at the beginning of every simulation step. This adaptive approach ensures the latest common operational picture is used to generate the target allocations. However, such a myopic approach may be limited due to its potential

sub-optimality over many time steps; study of improved methods is left for future work.

Once a UAV is given an allocation of a particular target, the UAV mode transitions to combat mode, and no longer adheres to the local behaviors of cohesion and separation. Other swarm members also exclude this UAV from further consideration that concern collective swarm motions. According to Day [8], UAV speed can be seen as a significant factor in UAV-versus-UAV combat. Therefore, when in combat mode and attempting to shoot down an enemy UAV, the UAV will accelerate up to v_Max in order to gain any tactical advantage possible. Additional characteristics of this combat mode are described in the following section.

2.2.5 Combat model

We assume that the UAV is equipped with a cookie cutter weapon that is pointing in the direction of $\vec{V}$ (i.e., relative to the nose of the UAV), and is able to hit the target within a range w with probability k. The weapons range, w, is assumed smaller than sensor range, s. The UAV i attacks the allocated target and tries to reduce the relative bearing angle, ω_{ij}, to the target j. ω_{ij} is the angle between $\vec{V}_i$ and $\vec{P}_{ij}$. As soon as $\omega_{ij} < 45/\left|\vec{P}_{ij}\right|$, and the range to the target is less than w, the UAV shoots at the target. The probability of kill is determined by k and the exposed surface area of target which depends on the angle ρ_{ij} between $\vec{P}_{ij}$ and $\vec{V}_j$. For that we see the target as a point and divide the anticipated sphere around the point in eight equal sectors. Each sector has an opening angle of 90 degrees and is associated with the front, rear, top, bottom, left or right side of the target. ρ_{ij} determines the sector and therefore the exposed area A_t. Then the actual kill probability is computed by

$$\pi = k\frac{A_s}{\max_s(A_s)}; \quad s \in \{1,2,3\}$$

Therefore an attacker has an higher chance to kill a target if it is attacking from below or above. We see the impact of this benefit related to the initial positioning. For example, a swarm that starts almost at the maximum ceiling, the main force attacks more likely from above and gets the tactical advantage, at least in the early phase of the combat. With this implementation, we hope to see configurations that exploit such considerations.

2.3 Markovian Model

This section outlines an analytical model to provide aligned but alternate insights about swarm tactics studied in this thesis. This stochastic process model describes the swarm engagement

scenario from a different point of view, and sensitivity analysis gives additional information about the consistency of the simulation results described previously.

We define a Markov process to explore the scenario in a mathematically rigorous manner. The presented approach explores two levels of abstraction. The first one deals with the one-on-one combat situation, whereas the second model applies the results of the first level to the multi-UAV scenario developed later on in this section. The decomposition into these two levels essentially takes out the individual dog fighting elements from the multi-UAV engagement. Studying both sub models provides more insights and allows for more tractable analyses. For each of these models, we are interested in the long term behavior of the Markov process, with absorbing states representing victory conditions for either Blue or Red swarms.

2.3.1 One-versus-One Combat

This model can be thought of as the engagement scenario that has only one UAV and the HVT on both sides. The UAVs start in a state in which they are not yet engaged in combat. There is some chance that an engagement (i.e., a dogfight) will ensue, in which one UAV shoots down the other or one UAV destroys the opposing HVT before it itself is defeated.

The following states describe the possible situation in such a scenario. Given that we do not consider fuel or ammunition shortages, there are no draw states; in other words, all end states dictate a clear win situation for one side or the other.

- **State** `0` (transient): Both UAVs start in this state. There is no combat at the beginning. This can also be envisioned to be the search phase where both try to detect the opposing UAV or fly against the adversary's HVT. Also, during an ongoing dogfight, there is some possibility that the evader is able to escape, at which point both UAVs return to this state `0`.
- **State** `BvR` (transient): Blue detects Red with some probability and manages to get in a pursuing position, or Blue was originally pursued by Red but has turned the table on Red. In this case, Blue has the tactical advantage and can try to kill Red.
- **State** `B` (absorbing): Blue pursued Red and shot Red down. The engagement is over and Blue won (since there is only one UAV on a side).
- **State** `RvB` (transient): Red detects blue with some probability and manages to get in a pursuing position, or red was originally pursued by blue and turned the table on blue. In this case, red has the tactical advantage and can try to down blue.

- **State** `R` (absorbing): Red pursued blue and shot blue down. The engagement is over and red won.
- **State** `HVTR` (absorbing): Blue reached red's HVT and destroyed it. This can happen even if Blue was previously engaged in a dogfight. Either red pursued blue and was not able to stop blue before it reached red's HVT, or blue attacked red and they came sufficiently close to red's HVT that blue had a higher reward to attack red's HVT than maintaining its pursuit of the Red UAV. In this case, Blue completed the mission and won.
- **State** `HVTB` (absorbing): Red reached blue's HVT and destroyed it. Red won in this case.

The state diagram for one-on-one combat is shown in Figure 2.8. Blue and red are together in one of the transient states `0`, `RvB` or `BvR` as long the scenario continues. As soon as one UAV enters one of the absorbing states, that is, `B`, `R`, `HVTR` or `HVTB`, the engagement is over and the corresponding party wins.

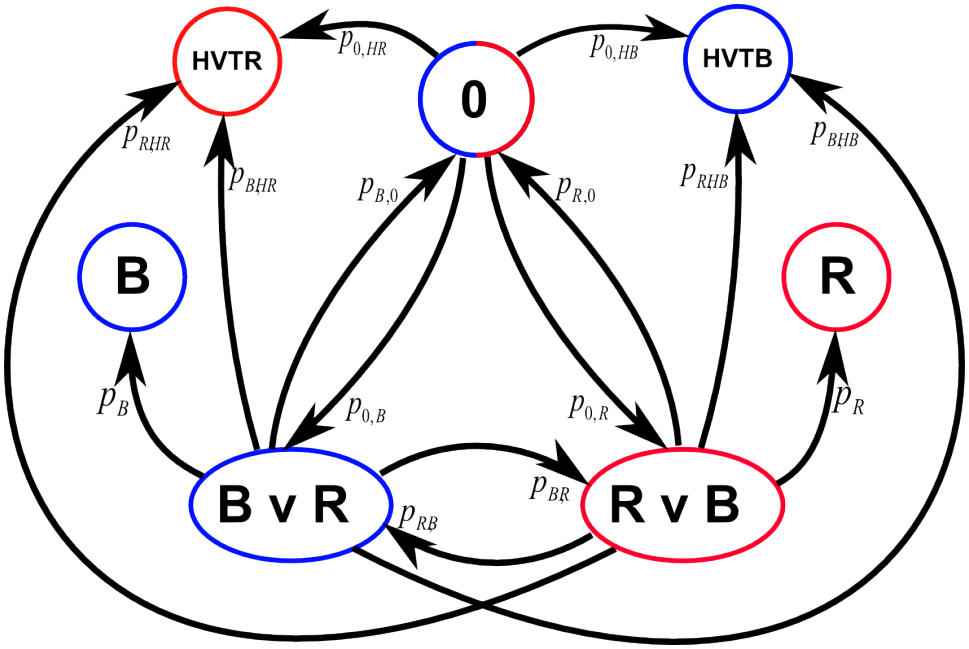


Figure 2.8: State diagram for the one-on-one combat (level 1).

For this state transition model, probabilities can be prescribed that govern the transition from one state to another, such that $p_{i,j}$ now represents the probability of transitioning from state i to state j. The transition probability matrix can be written in canonical form, given by

$$\begin{pmatrix} I & 0 \\ R & Q \end{pmatrix}$$

where R gives the transition from a transient state to an absorbing state and Q represents the transitions among the transient states. Table 2.3.1 thus is the matrix representation in this canonical form of the Markov process illustrated in Figure 2.8.

	B	R	HVTR	HVTB	0	BvR	RvB
B	1						
R		1					
HVTR			1				
HVTB				1			
0	0	0	$p_{0,HR}$	$p_{0,HB}$	0	$p_{0,B}$	$p_{0,R}$
BvR	p_B	0	$p_{B,HR}$	$p_{B,HB}$	$p_{B,0}$	0	$p_{B,R}$
RvB	0	p_R	$p_{R,HR}$	$p_{R,HB}$	$p_{R,0}$	$p_{R,B}$	0

Table 2.2: One-step transition probability matrix for level one (one-on-one combat)

Given this canonical form, the steady-state transition probabilities from the transient states to the absorbing states can readily be computed by the matrix equation, as referenced in Ross [24]:

$$B = (I - Q)^{-1} R \tag{2.1}$$

Since the engagement always starts in state `0`, we are only interested in the transition probabilities from this state into the absorbing states, summarized by the following matrix:

$$B^1 = \begin{pmatrix} p^1_{0,B} & p^1_{0,R} & p^1_{0,HR} & p^1_{0,HB} \\ p^1_{B,B} & p^1_{B,R} & p^1_{B,HR} & p^1_{B,HB} \\ p^1_{R,B} & p^1_{R,R} & p^1_{R,HR} & p^1_{R,HB} \end{pmatrix}$$

These transition probabilities represent the starting point for the second level model, moving beyond one-on-one combat and describing multi-UAV engagements.

2.3.2 Multi-UAV Engagement

To study swarm engagements, we want to increase the numbers of UAVs on each side to 50 as described in the original reference scenario. In order to do so, we assume that only one event happens at a distinct point in time. This is either the defeat of a UAV or the destruction of an HVT. The results from the previous section determine the probability at which each event occurs. A UAV gets shot down with probability $p^1_{0,B}$ or $p^1_{0,R}$, and an HVT is taken out of action with probability $p^1_{0,HR}$ or $p^1_{0,HB}$.

The transient states for this Markov process are represented by the available UAVs on both sides. The state `50/50` is the starting point where 50 blue UAVs and 50 red UAVs are in the battle arena. In this naming convention, the first number always reflects the number of blue UAVs and the second number counts the red UAVs. Starting with `50/50`, the stochastic process either goes to `50/49` with probability $p^1_{0,B}$ or to `49/50` with probability $p^1_{0,HB}$. This process goes on until one side reaches zero. All states where one party has zero UAVs left is an absorbing state. In every transient state, there is also some probability that one of the HVTs get destroyed, in which case, the engagement is determined, and the party who destroyed the HVT won. For an illustrate of this multi-UAV engagement Markov model, see Figure 2.9.

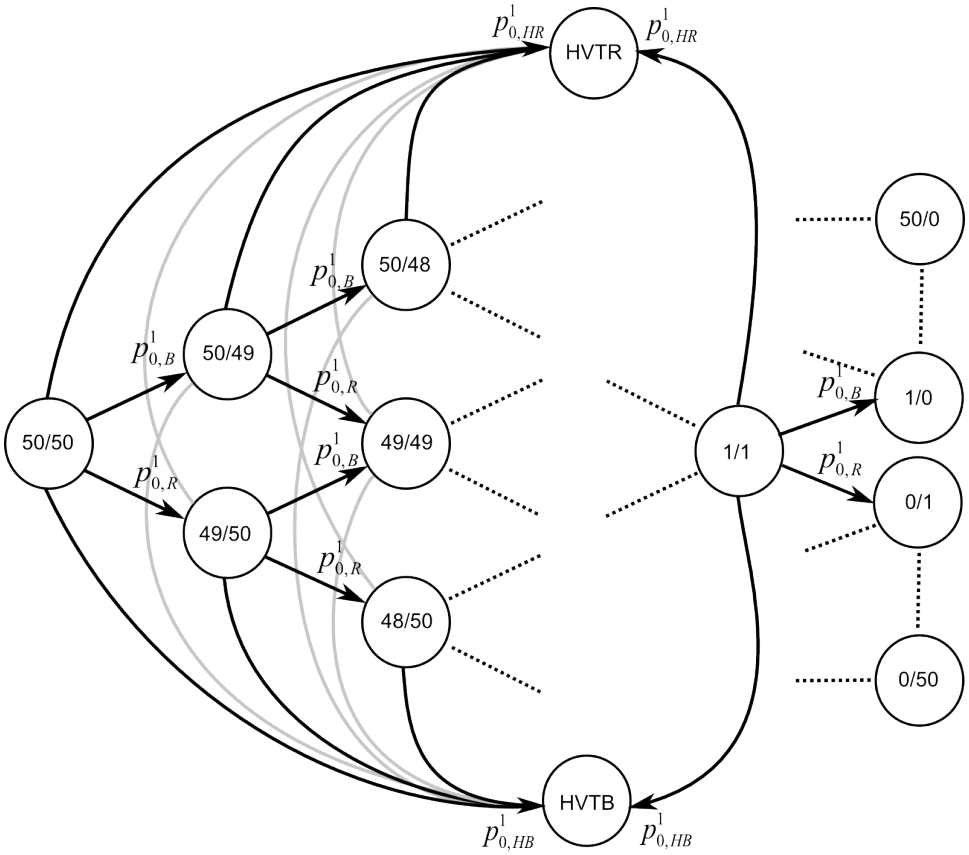


Figure 2.9: State diagram for level two (multi-UAV engagement).

As before, we wish to study the long term behavior of such a multi-UAV engagement. We once again use the canonical form of the transition probability matrix as before (see Equation 2.1). Then the relevant transition probabilities for this second level model are summarized by B^2:

$$B^2 = \begin{pmatrix} p^2_{50,50/0} & \cdots & p^2_{50,1/0} & p^2_{50,0/1} & \cdots & p^2_{50,0/50} & p^2_{50,HR} & p^2_{50,HB} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\ p^2_{1,50/0} & \cdots & p^2_{1,1/0} & p^2_{1,0/1} & \cdots & p^2_{1,0/50} & p^2_{1,HR} & p^2_{1,HB} \end{pmatrix},$$

which determines the long term transition probabilities from any transient state into all absorbing states. Since the scenario starts always with 50 blue against 50 red UAVs, the first row of B^2 gives the probabilities for the different end states. For completeness, the full transition matrix corresponding to this Markov model is thus given in Table 2.3.2.

	50/0	49/0	$\cdots$	1/0	0/1	$\cdots$	0/50	HVTR	HVTB	50/50	50/49	$\cdots$	49/50	49/49	$\cdots$	1/1
50/0	1															
49/0		1														
$\vdots$			$\ddots$					0								
1/0				1									0			
0/1					1											
$\vdots$		0				$\ddots$										
0/50							1									
HVTR								1								
HVTB									1							
50/50	0	0	$\cdots$	0	0	$\cdots$	0	$p^1_{0,HR}$	$p^1_{0,HB}$	0	$p^1_{0,B}$	$\cdots$	$p^1_{0,R}$	0	$\cdots$	0
50/49	0	0	$\cdots$	0	0	$\cdots$	0	$p^1_{0,HR}$	$p^1_{0,HB}$	0	0	$\cdots$	0	$p^1_{0,R}$	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
49/50	0	0	$\cdots$	0	0	$\cdots$	0	$p^1_{0,HR}$	$p^1_{0,HB}$	0	0	$\cdots$	0	$p^1_{0,B}$	$\cdots$	0
49/49	0	0	$\cdots$	0	0	$\cdots$	0	$p^1_{0,HR}$	$p^1_{0,HB}$	0	0	$\cdots$	0	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
1/1	0	0	$\cdots$	$p^1_{0,B}$	$p^1_{0,R}$	$\cdots$	0	$p^1_{0,HR}$	$p^1_{0,HB}$	0	0	$\cdots$	0	0	$\cdots$	0

Table 2.3: One-step transition probability matrix for level two (multi-UAV engagement).

Given this theoretical model formulation, further analysis in Chapter 4 will outline how we change some of the probabilities in the level one model and observe its impact on steady state probabilities in the second level model. The sensitivity analysis should disclose critical parts in the scenario and enable recommendations for systems in such an environment.

CHAPTER 3:
Design of Experiments

After having defined and developed the two complementary models for exploring the swarm-versus-swarm scenario, we now want to use these models to gain insights regarding swarm tactics in swarm combat engagements. In particular, we design simulation experiments for the constructed simulation model, using two different experimental designs to address the complexities of the model. The objective of the experimental design described in this chapter is to discover and characterize the influential variables and their interactions. Knowledge of these significant factors directly guides the rigorous statistical analysis conducted in Chapter 4, in which we delve more deeply into the results of the simulation experiments, and also re-connect those results with the Markov process stochastic modeling effort.

In terms of the applied scenario, swarm combat could potentially be broken down into multiple one-on-one fights. In this case, parameters such as speed, maneuverability, or weapon capabilities would be decisive. However, the purpose of the present study is to specifically investigate the implications of swarming with large numbers of UAVs on each side. We reiterate this emphasis as the reason for assuming technological parity by using the same type of UAV on both sides. In this context, a dogfight between two singular UAVs would always be undetermined since both are equally matched. Instead, other tactical factors like initial positioning, offense-defense preferences, or swarm behavior may influence the simulation outcome, and their illumination is the focus of this thesis.

During the development phase of the simulation model, we incorporated a preliminary intuition on the nature of the swarm engagement scenario by identifying many design parameters relevant to the scenario and combat environment, the individual agent motion and sensing rules, and the collective swarm behaviors. In this chapter, we more formally describe the variables of interest and describe the produced simulation output. We then outline and generate various statistical experimental designs, which help to screen for important regressors and interactions between the defined input variables. Having also defined the appropriate response variables, namely the measures of effectiveness including the probability that Blue wins the engagement and the surviving force strengths of Blue and Red UAVs, the simulation experiments themselves and relevant quantities are described in further detail.

3.1 Variables of Interest

Based on preliminary assessment of the simulation, we identify twelve key variables which seem to influence the responses. These variables and their nominal ranges are provided in Table 3.1, with more detailed description of each of the relevant factors given thereafter.

Variable	Name	Range
X_1	NumAllocBlue	[1, 9]
X_2	NumAllocRed	[1, 9]
X_3	CohesionMax	[500, 10000]
X_4	ConvergeBlue	[500, 10000]
X_5	ConvergeRed	[500,10000]
X_6	WeightBlue	[0, 10]
X_7	WeightRed	[0, 10]
X_8	StayingPwHVT	[1, 11]
X_9	YAlpha	[1, 1001]
X_{10}	YBeta	[1, 1001]
X_{11}	ZAlpha	[1, 1001]
X_{12}	ZBeta	[1, 1001]

Table 3.1: Simulation variables considered in the presented statistical design of experiments. Each of these factors are systematically varied over the defined ranges during several simulation replications.[1]

NumAllocBlue **and** NumAllocRed are the maximum numbers of possible allocations of friendly UAVs to a single detected (enemy) target. The number of allocations ranges from one to nine. This range seems reasonable since we have to allow at least one allocation and even if ten allocations are possible, this number will likely rarely be observed because of the penalty function for target allocation explained in Section 2.2.4. However, given the possibility that there are isolated situations where a whole swarm on one side meets just a few opposing UAVs, this high factor level may identify interesting results. In general, we can think about this variable as a measurement of force concentration at the engagement level. A higher value will concentrate the fire on few enemies with a higher survival rate for the friendly force.

CohesionMax determines the distance relative to a given UAV within which it includes swarm members in its calculation for the center of mass. This parameter influences the individual UAV's behavior for cohesion, which guides the collective motion, and gives an idea of how close the cohesion within the swarm is. The range goes from 500 meters to 10000 meters,

[1]The ranges are adjusted (e.g., upper levels at 9, 11 or 1001, instead of 10 and 1000, respectively) such that the center values within the interval are integers (e.g., 5, 6 and 500, vice 4.5, 5.5, and 500.5).

where the swarm is loosely assembled for lower values and transits in more tightly arranged configurations for higher values.

`ConvergeBlue` **and** `ConvergeRed` define the distance to the respective adversary's HVT at which the swarm UAVs converge to attack the HVT directly. A small value of 500 meters means that the swarm uses the whole width of the battle space to approach the area where the opposing HVT is located and attack it from three sides. In the case where the enemy sends its UAVs straight through the middle, employing a short convergence range might allow friendly UAVs to reach the HVT without many engagements. This would amount to a tactic known as a flank attack. The trade-off in this scenario is that both sides have the same number of UAVs. Therefore, if executing a flank attack, then the center is nearly undefended against an enemy's concentrated force down the middle, and approaching from the flanks results in a longer path to the HVT. Alternatively, for example, with a high value of 10000 meters for this parameter, the swarm would start their attack directly at the beginning and nominally concentrate their force in the center. This could mean that some opposing UAVs could reach the rear area undiscovered behind the advancing friendly swarm while the swarm is forced into dogfights with the adversary's main force and delayed from striking the adversary's HVT.

`WeightBlue` **and** `WeightRed` determine the weight factor f in the penalty function introduced in Section 2.2.4. In general, this value defines whether a swarm is more or less defensive or offensive. A low value of zero leads to a lack of preference between attacking the adversary's HVT and attacking a detected enemy UAV, with the friendly UAV attacking whichever is opportunistically closer. In this case, if the two swarms meet in the open battle space, the distance to every opposing UAV is smaller than to the opposing HVT. Therefore, a UAV always attacks detected targets and tries to prevent them from attacking its own HVT. We see this as a defensive posture. As the value of f increases, the preference of the UAV to attack the HVT gets stronger. With a value of 10, the UAV no longer cares about engaging any enemy UAVs but rather is focused on attacking the HVT. We can interpret this as offensive behavior. Let us look at a simple arithmetic example. Suppose the weight factor is nine. The battle space has a dimension of ten by ten kilometers. Therefore, the distance from the UAV to the opposing HVT is about ten kilometers at the beginning. Both swarms meet approximately in the middle. So, the distance to the HVT is approximately six kilometers. A target is detected 0.7 kilometers away. But $0.7 \cdot 9 > 6$ and the UAV ignores the detected target.

`StayingPwHVT` defines the number of hits needed in order to take an HVT out of action. This value was originally not supposed to be a variable for this analysis. Early simulation runs showed that it is easy for a swarm to have at least one successful UAV strike on the HVT. However, this result in isolation does not mean that the swarm necessarily has a tactical advantage. It is also more realistic to assume that an HVT is able to resist more than one attack, either through design or by self-defense capabilities.

`YAlpha`**,** `YBeta`**,** `ZAlpha` **and** `ZBeta` are the input parameters for the two Beta distributions which generate the initial positioning for the Blue swarm. We want to explore if the initial spatial configuration of the swarm makes a difference in the outcome of a battle while the adversary swarm is always uniformly distributed. Rather than defining fixed setups, we would like to screen over all possible initial positions. The two-parameter Beta distribution is one option to achieve this goal. There are two main advantages of this distribution. First, it always produces random numbers between zero and one, which can easily be scaled to the dimensions of the battle space. Second, varying the input parameters, α and β, structurally changes the output distribution, spanning from the uniform to unimodal to left- and right-skewed distributions. If we do so and vary the four input parameters (two each for y- and z-axes), we see spatial configurations that are (a) almost equally distributed over the whole space, (b) more centered, (c) more concentrated on the left, right or (d) on both flanks, and analogously with changes in the altitude. Figure 3.1 shows how the input parameters influence the positioning. The prescribed ranges for these parameters is $[1, 1001]$. Note that this interval is only for convenience while we generate the design, as the simulation divides each value by 100, such that the effective range that the simulation uses is $[0.01, 10.01]$.

3.2 Generation of the Design of Experiments

The goal is to explore the above mentioned factors and their two- and three-way interactions with respect to their influence on the simulation output. We also must check for non-linearity and, if necessary, adapt the model subsequently in the analysis. One widely used approach in design of experiments (DoE) are factorial designs (see [25]) where every factor is observed at its low and high levels, and possibly one or more levels in between. These designs are orthogonal, which results in no correlation between all the main effects and their interactions. They also provide the ability to explore all possible interactions. The biggest drawback is the high number of design points that have to be run. In our study, we have twelve factors, and we recognize the need for at least three levels per factor because we want to observe any nonlinear effects. Such a

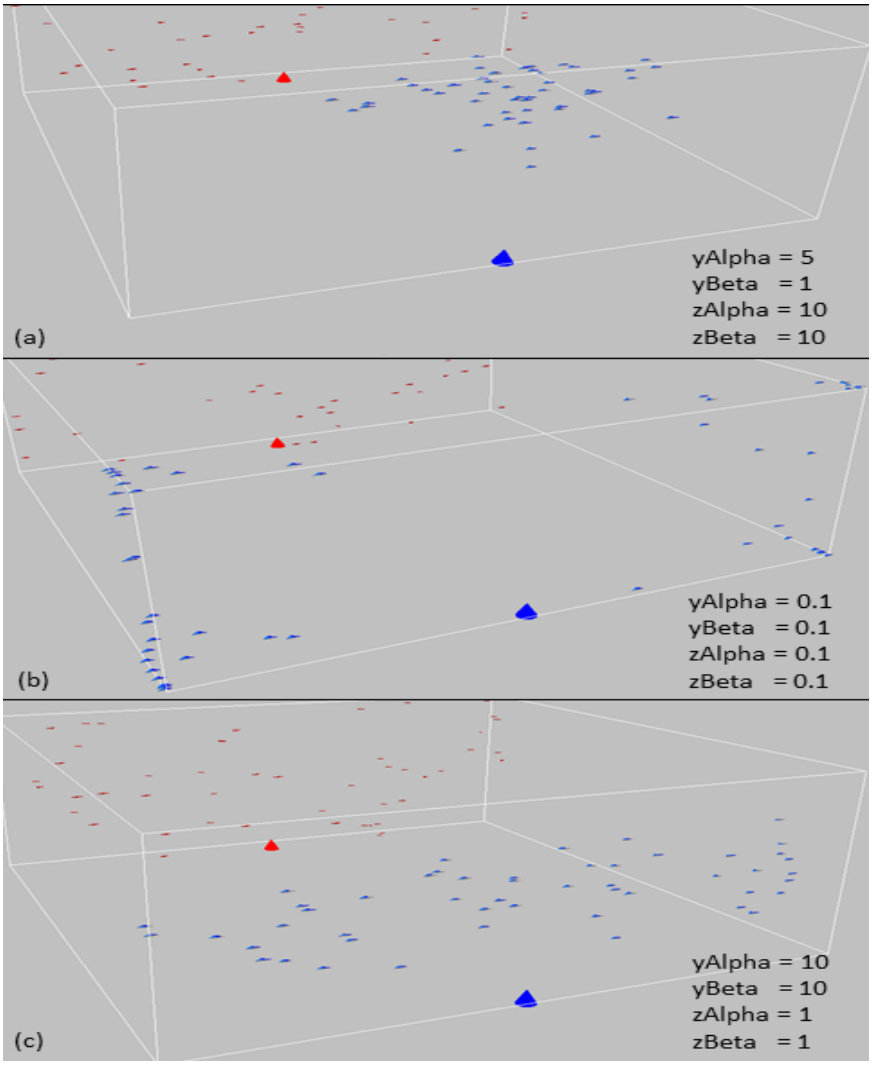


Figure 3.1: Illustrations of various Blue UAV swarm initial position configurations while varying the annotated input parameters for the beta distribution in y- and z-directions, yielding (a) a more centralized setup in a higher position, (b) a force concentration on both flanks, and (c) a uniform distribution in the width and centralization in the height.

design would lead to $3^{12} = 531441$ design points. Let us say we run every design point with 30 replications, which now results in $15,943,230$ total replications. Given that a single run takes about four seconds on an up-to-date computer, we would need about *two years* to complete the experiment. This is not acceptable.

An alternative to factorial designs are fractional factorial designs. These designs assume that higher order terms are essentially negligible. The resolution of such a design explains the way in which the observed effects are confounded with each other. For example, a resolution-III

fractional factorial design ensures that the main effects are not confounded with each other; however, they may be confounded with two-way interactions or other higher-order terms. If we were to use such a design, we would be restricted to the main effects only, and would be forced to assume that higher-order terms are noise that do not influence the model significantly. Given the variability and complexity of the model, we wish neither to make this assumption nor to discard potentially meaningful influences. According to the stated goal above, a resolution-VII fractional factorial design appears to be a satisfactory choice: The main effects include only the noise from the sixth-order terms or higher, the second-order terms only from the fifth-order terms or higher, and the third-order terms from the fourth-order terms or higher. In other words, we assume that the fourth-order terms and higher are not significant for our model. For a further discussion of fractional factorial designs, we refer the reader to Sanchez [25] or Kleijnen et al. [26]. Draper and Lin [27] state that a resolution-VII fractional factorial design for twelve factors (2_{VII}^{15-5}) would comprise 2^{10} design points. If we repeat the calculation as above, then we need only $30,720$ runs. This is approximately equivalent to one and a half days, and if we were to take advantage of multi-core processing, then we could decrease this to an overnight run, which is more appropriate for our purposes.

However, this choice of design would not completely achieve the designated goal, since the fractional factorial designs observe only two levels of each factor. This high-low setup would be enough for a linear regression model, but we would not be able to see any nonlinear effects if some are present. However, a center composite design (CCD) is one that we can use to observe non-linearity in the model [25]. We take the 2_{VII}^{15-5} design and add center point and star-points for every factor. A star point varies only one factor at a time. All other factors are kept at the center of the range. Therefore, we see two star points for every factor (one for the low and one for the high value). The resulting design contains 1050 design points and gives us the opportunity to fit a model with three-way interactions and to observe nonlinear effects. Sanchez [28] provides a solution to generate such a design effectively (see Table 3.2), where the factors, $X_1, \ldots, X_{12}$ are the variables in the same order as explained in Section 3.1 (also refer to Table 3.1).

As we will see in later analysis, there are non-linear effects among the factors. So we did well by choosing a design that is able to detect such effects. However, with only three observed levels, we cannot fully explore the real nature of these effects and not all are necessarily quadratic. We would need more observations in between at intermediate levels. A Latin hypercube design is useful in this case because of the space-filling properties and the relatively few design points

n	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}
1	1	1	500	500	500	0	0	1	1	1	1	1
2	9	1	500	500	500	0	10	1	1	1	1001	1
3	1	9	500	500	500	0	10	1	1	1	1001	1
4	9	9	500	500	500	0	0	1	1	1	1	1
5	1	1	10000	500	500	0	10	1	1	1	1001	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
531	1	9	500	500	10000	0	0	1	1	1	1001	1001
532	9	9	500	500	10000	0	10	1	1	1	1	1001
533	1	1	10000	500	10000	0	0	1	1	1	1001	1001
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1023	1	9	10000	10000	10000	10	10	11	1001	1001	1001	1001
1024	9	9	10000	10000	10000	10	0	11	1001	1001	1	1001
1025	5	5	5250	5250	5250	5	5	6	501	501	501	501
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1048	5	5	5250	5250	5250	5	5	6	501	501	1	501
1049	5	5	5250	5250	5250	5	5	6	501	501	501	1001
1050	5	5	5250	5250	5250	5	5	6	501	501	501	1

Table 3.2: Resolution VII fractional factorial design with star and center points for 12 factors. X_1 to X_{12} refer to the variable names in Table 3.1. There are 1050 design points overall. [2]

needed. Sanchez provides a spreadsheet tool that generates Nearly Orthogonal Latin Hypercube (NOLH) designs for up to 29 factors [29]. This tool satisfies our need for 12 factors, and the resulting NOLH design comprises 65 design points. This design is shown in Table 3.3.

3.3 Simulation Response Variables

In the sections above, we outline and design the input parameters for the simulation model. We now turn to the simulation output, which represent the measures of effectiveness or response variables that we analyze in the next chapter to make recommendations for successful swarm behaviors and generation of swarm tactics. As they will be addressed in thorough detail, we simply state and briefly describe each of the simulation outputs here.

`BlueWon` is a binary variable that is zero if Red was successful in a particular simulation run, and one if Blue accomplished the mission, as defined by destroying the adversary's HVT. Several runs are be conducted for each design point. The average of this response variable, `BlueWon`,

[2]The complete design matrix in comma-separated value format (filename `DoE_CCD.csv`) can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

n	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}
1	7	1	3914	3617	1688	8	8	5	969	719	547	938
2	10	7	1539	4508	3766	3	5	8	719	922	766	485
3	9	4	9555	2578	3320	9	2	5	407	579	797	875
4	7	9	7328	4805	1094	4	3	3	126	984	906	625
5	9	5	2281	648	1391	2	3	6	610	485	63	844
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
61	2	2	4063	2133	1242	7	9	4	266	641	625	17
62	2	7	8813	4211	3914	8	6	9	781	141	126	454
63	4	2	6734	3023	1539	0	7	5	938	220	391	329
64	4	9	3320	4656	945	6	2	3	79	32	188	469
65	2	4	1094	1836	3172	4	1	4	344	391	173	235

Table 3.3: Nearly Orthogonal Latin Hypercube design for 12 factors. X_1 to X_{12} refer to the variable names in Table 3.1. There are 65 design points overall. [3]

for a particular set of runs gives the probability for an expected Blue success for a particular design point. Higher probabilities identify better configurations for the Blue swarm and indicate potentially relevant swarm tactics considerations.

`SurvRed` is the number of Red survivors. Even if Blue accomplished the mission and destroyed the Red home base, Red could have more UAVs remaining than Blue. These enemy UAVs would nominally continue to pose a threat on Blue's home base. This fact should also be taken into consideration when developing simulation configurations. This output is also a starting point for further development of the simulation model to modify the termination criterion for multiple HVTs or extended engagements.

`SurvBlue` is the number of Blue survivors. The same statements above hold for this measure. We could combine `SurvRed` and `SurvBlue` as a force ratio and incorporate this ratio into the analysis as a single response variable.

`DistanceCenter` **and** `DistanceUAV` are not explicitly MOEs; however, these variables are generated by the simulation and are closely connected to `YAlpha`, `YBeta`, `ZAlpha` and `ZBeta` parameters. Though the α and β values for the beta distributions in y- and z-axis initial positions have no directly operational meaning, we can transform these parameters into measurements that can be taken in operational contexts which express the initial positioning of swarm quite well. `DistanceCenter` gives the distance from the center of the bin where the Blue swarm

[3]The complete design matrix in comma-separated value format (filename `DoE_NOLH.csv`) can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

is initially staged to the COM of the entire swarm. This value gives an idea of how the main force of the swarm is situated with respect to the HVT, recalling that the center of the bin is directly above the HVT. `DistanceUAV` is the average distance of the UAVs to the COM of the swarm. The higher the value, the more dispersed the swarm is initially. Figure 3.2 illustrates this visually. Rather than using the α and β values as regressors in the simulation analysis, we choose to analyze the more operationally grounded responses of `DistanceCenter` and `DistanceUAV`.

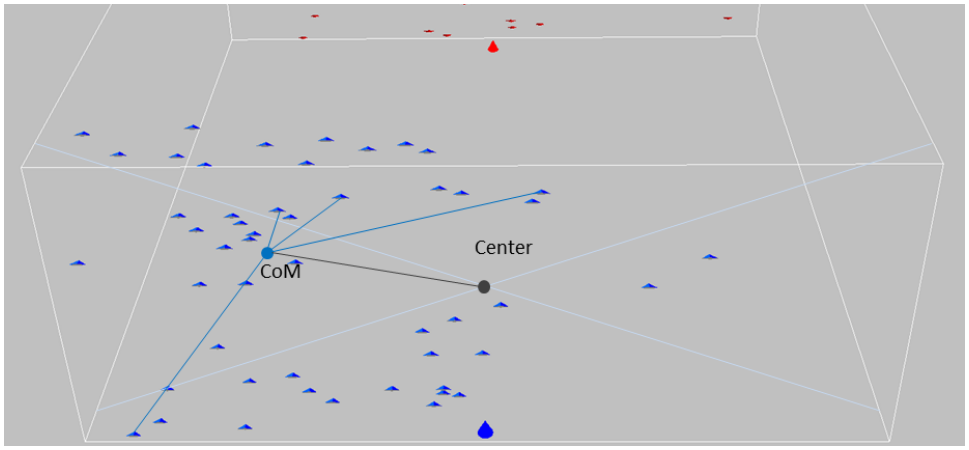


Figure 3.2: Schematic diagram for the measurement of response variables, DistanceCenter and DistanceUAV. "Center" is the center of the bin where the Blue swarm is positioned initially. "COM" is the center of mass of the swarm. `DistanceCenter` is the gray line that describes the distance between "Center" and "COM." `DistanceUAV` is the average distance of all UAVs to the COM (symbolized by the blue line).

3.4 Conducting the Experiment

We wish to run the simulation with the generated designs. Before we can begin execution, we must find the appropriate number of replications for each design point to achieve the statistical power we require. A power calculation can help here and is essentially an inverted hypothesis test where we specify the sensitivity level and the probabilities for Type I and Type II errors at this specified level. Let us state the hypothesis first.

$$H_0 : p_{BlueWon} = 0.5$$
$$H_a : p_{BlueWon} \neq 0.5$$

The null hypothesis is that Blue has a 50% chance to win for every given design point. The alternative is that Blue's probability of a success is lower or higher than 50%. So, this is a

two-sided test. Remember that we use a central composite design which enables screening for important factors. Though it will not capture or explain the whole model, we consider this initial exploration as acceptable as we conduct this preliminary analysis to obtain some early insights. We should account for this preliminary nature in our determination of an appropriate number of replications. In our study, the Pareto principle can be applied to this case, which says that we can explain 80% of the model behavior with 20% of the effort (as measured by the number of simulation runs in this case) that we would need to explain 100% of the model. We can set the probabilities for Type I and II error according to this principle. We reject H_0 with a maximal error of $\alpha = 0.2$ (Type I error) and fail to reject if H_a is true with $\beta = 0.2$ (Type II error) at a sensitivity level of 0.1. As this is a two-sided test, a sensitivity level of 0.1 is equivalent to an interval of 20% centered about the mean. The number of replications that gives the desired power is determined by:

$$n = \left(\frac{\sigma(z_{\alpha/2} + z_{\beta})}{(p_a - p_0)} \right)^2,$$

where $(p_a - p_0)$ defines the sensitivity level, denoted d. For those who are interested in a deeper discussion about this part, we recommend Wackerly, Mendenhall and Scheaffer [30]. As we do not know the true standard deviation σ of the population, we can estimate it by running the simulation with an appropriate initial number of replications. The presented NOLH design is a good choice for this purpose because of the fewer design points compared to the central composite design. We run this design with ten replications and calculate s_i, the standard deviation for design point $i \in (1, ..., 65)$. Then

$$n_i = \left(\frac{s_i(z_{\alpha/2} + z_{\beta})}{d} \right)^2$$

is the number of runs needed to reach the threshold for design point i. Observe that the s_i are not equal; therefore, the suggested n vary with design point as well. Hence, we find

$$n^* = \max(n_1, ..., n_{65})$$

as the maximum number of replications among all design points, and we use this number for the entire experiment for all design points. The computed n_i are found to be in the range, $[0, 126]$. Hence, we determine that we need $n^* = 126$ replications for each design point.

To further refine our experiment setup, Law [31] mentions a popular variance reduction tech-

nique called common random numbers (CRN) for experiments that compare two or more system configurations. We use this technique in our case, too. The simulation is nominally restarted at every design point and is executed for the defined number of replications (in this case, 126). Once all replications are completed, the simulations end for that design point and starts over with the next configuration. Typically, the current system's CPU time stamp is commonly used as the starting seed for the first run, incremented by one for the next replication and so on. Law instead suggests taking the same seed for the first replication of every design point. This forces the simulation to use the same random numbers for each respective configuration. At the end, we see a positive correlation between the design points, which reduces the variance.

Upon completion of these design specifications, we can conduct the simulation experiments with the defined designs. The simulation reads a CSV-file with the design points and runs every configuration with $n^* = 126$ replications. Figure 3.3 illustrates different snapshots in the evolution of the simulation for one particular replication. At the conclusion of the run, the program appends the aforementioned simulation responses to the same CSV-file. Table 3.4 gives an example for the NOLH design with ten replications. The variables X_1 to X_{12} are the original input parameters, Y_1 to Y_3 are the simulation responses, and X_{13} and X_{14} represent the computed surrogate factors, `DistanceCenter` and `DistanceUAV`, respectively.

n	`config`	X_1	$\cdots$	X_{12}	Y_1	Y_2	Y_3	X_{13}	X_{14}
1	1	7	$\cdots$	938	36	37	1	1367	1045
2	1	7	$\cdots$	938	46	35	0	1347	1100
3	1	7	$\cdots$	938	45	38	0	1271	1207
4	1	7	$\cdots$	938	39	36	0	1501	1212
5	1	7	$\cdots$	938	46	40	0	1319	1114
6	1	7	$\cdots$	938	36	41	1	1430	1106
7	1	7	$\cdots$	938	46	36	0	907	1029
8	1	7	$\cdots$	938	44	41	0	1291	1063
9	1	7	$\cdots$	938	45	43	0	1298	1145
10	1	7	$\cdots$	938	43	38	0	1434	1244
11	2	10	$\cdots$	485	33	28	1	1359	1233
12	2	10	$\cdots$	485	29	21	1	1001	1152
13	2	10	$\cdots$	485	32	28	1	807	1124
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 3.4: Example simulation output for ten replications on each design point where Y_1 corresponds to SurvBlue, Y_2 to SurvRed, Y_3 to BlueWon, X_{13} to DistanceCenter and X_{14} to DistanceUAV. n is the run number and `config` identifies the design point.

The CSV-files with the simulation output for the CCD (`resultDoE_CCD.csv`) and the NOLH

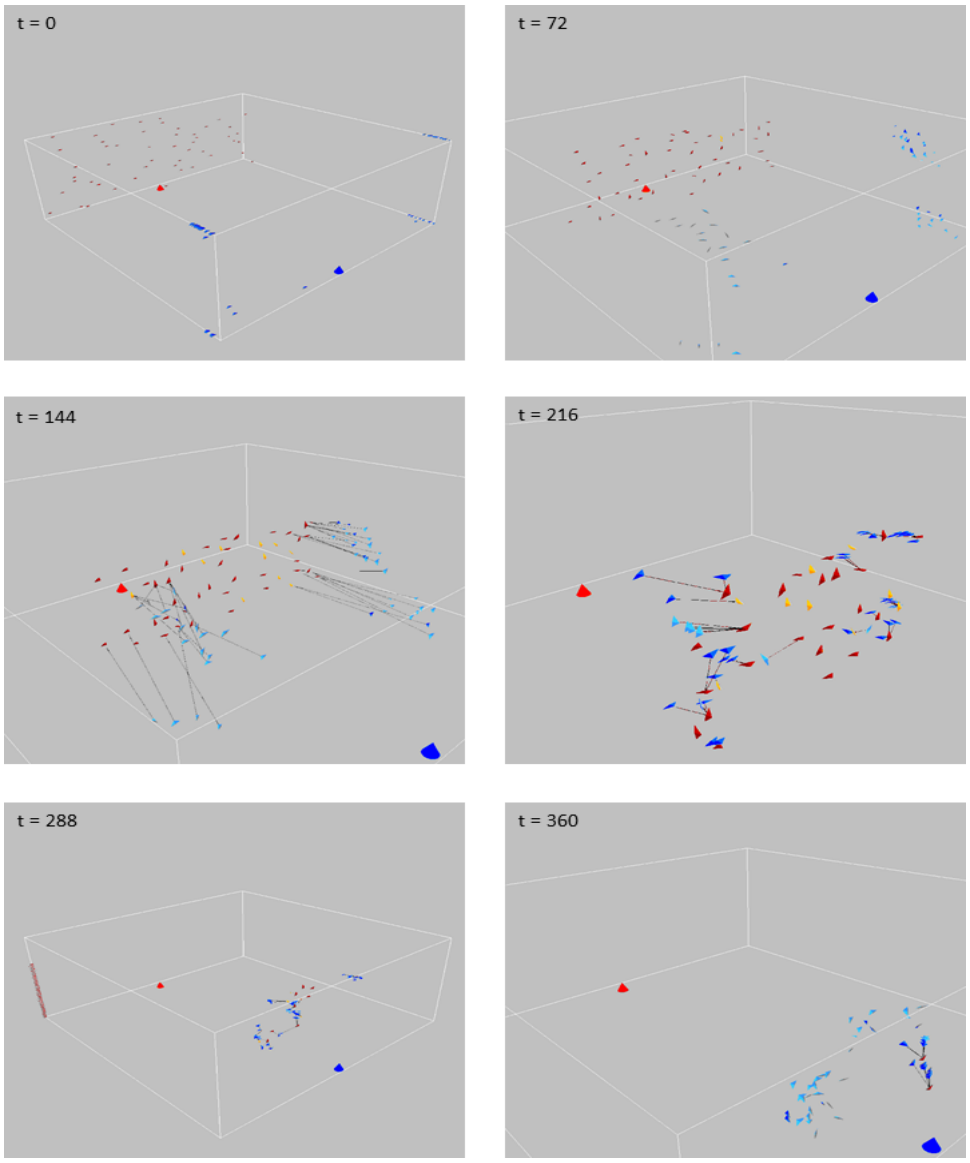


Figure 3.3: Run number 13 of the central composite design at time t = 0, t = 72, t = 144, t = 216, t = 288 and t = 360.

design (`resultDoE_NOLH.csv`) can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

CHAPTER 4:
Analysis

Having formulated and developed the models of interest, designed the experiment, and generated response data, this chapter describes the methods of analysis used to generate insights on UAV swarm tactics. We first perform the analysis on the data from the agent-based Monte Carlo simulation model, followed by exploration of and comparison with the Markov process and its analytic properties. These studies identify numerous insights relevant to future modeling efforts as well as for generation of swarm tactics.

4.1 Simulation Model Analysis

We concentrate the analysis on one of the three measures provided by the simulation, namely `BlueWon`, which represents the binary response that is either zero if Red wins or one if Blue wins. Accordingly, `BlueWon` is a binomial random variable at each design point, such that Blue has 126 trials and the number of successes are counted. The bounded responses (from zero to one hundred percent) merit the use of logistic regression in this analysis. This type of regression model belongs to the group of generalized linear models (GLM) and primarily rests on three elements:

- The response is assumed independently distributed with a distribution that belongs to the exponential family;
- A linear predictor with k variables ($k = 10$ in this case); and
- A link function.

The design of experiments and the simulation model ensures that `BlueWon` is independently binomially distributed (part of the exponential family) and the `Logit` link function serves as the right choice in our case. However, recall that we indicated in Section 3.2 that we have non-linearity in the main effects. So, before we can fit a GLM, we must address this issue and transform the regressors, as appropriate. We would like to perform this transformation with the dataset generated by the CCD; however, with only three levels for each factor, we are only able to recognize variables expressing non-linearity but not explore their nonlinear nature. To facilitate this deeper analysis, we use the dataset produced by the NOLH design.

4.1.1 Assessing the Raw Response Data

Before we start with the analysis, we take a look at the data and especially at the the responses. We can use the data, particularly the responses, from the CCD, as this design is used to fit the initial model. Let us first conduct the transformation of parameters relevant to the initial spatial positioning of the Blue swarm (namely, `yAlpha`, `yBeta`, `zAlpha` and `zBeta`) into surrogate parameters, `DistanceCenter` and `DistanceUAV`, representing more tactically relevant descriptions. Both regressors are shown in one plot in Figure 4.1. Observe that we do not fill the space due to the nature of the fractional factorial design, as expected. However, we see a diversity of initial positions represented. For example, low `DistanceCenter` and low `DistanceUAV` give a configuration representing an amassing of forces centrally in the airspace, low `DistanceCenter` and high `DistanceUAV` provide a positioning where the main force is split on both flanks, and high `DistanceCenter` and low `DistanceUAV` concentrates the UAV swarm initially on one flank.

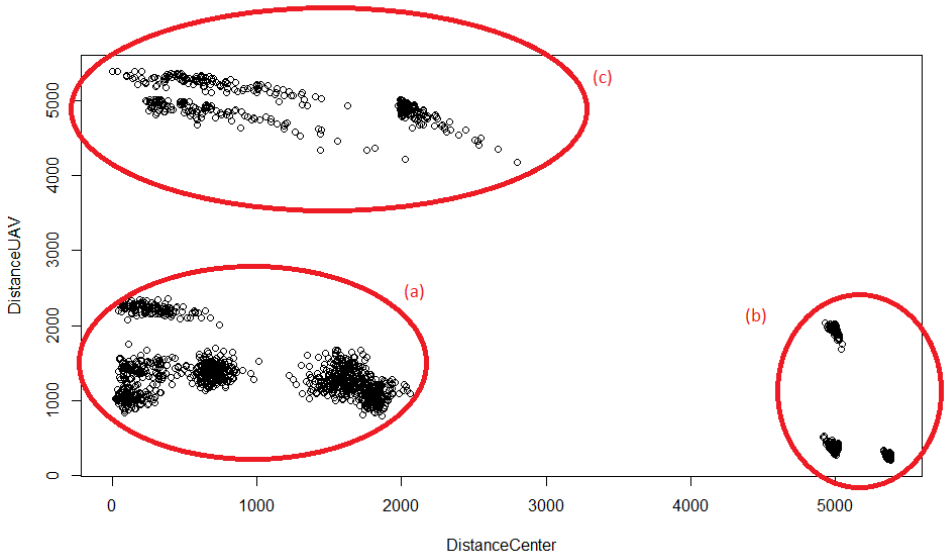


Figure 4.1: Scatter plot showing `DistanceCenter` against `DistanceUAV` design points. The central composite experimental design can be seen to not be space filling; however, the clusters roughly outline different tactical configurations, namely (a) concentrated forces in the center of the field, (b) positioning on one flank only and (c) strong flanks, that is, forces amassed on both flanks.

There will never be data points with high `DistanceCenter` and high `DistanceUAV` because high `DistanceCenter` requires all UAVs to be together on one edge but this reduces the distance between the UAVs. So, we will never observe a balanced design for these two factors.

As we refer to Figure 4.2, we see that all regressors have a centered median except `DistanceCenter`

and DistanceUAV. The CCD is not space filling like an NOLH design but it still spreads the regressors evenly over the whole range. Unfortunately, we do not find this to be the case for DistanceCenter, which is nearly but not quite centered, and DistanceUAV, which is far off. Nonetheless, these two derived variables provide more meaningful explanation of the tactical picture than the original α and β values as descriptors of the initial spatial configuration of the swarm.

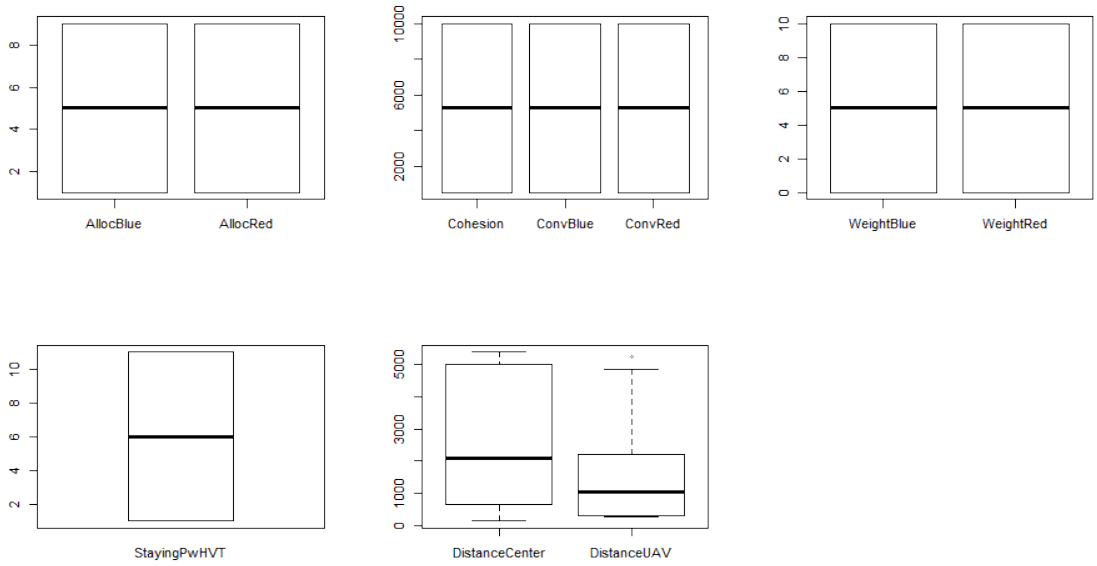
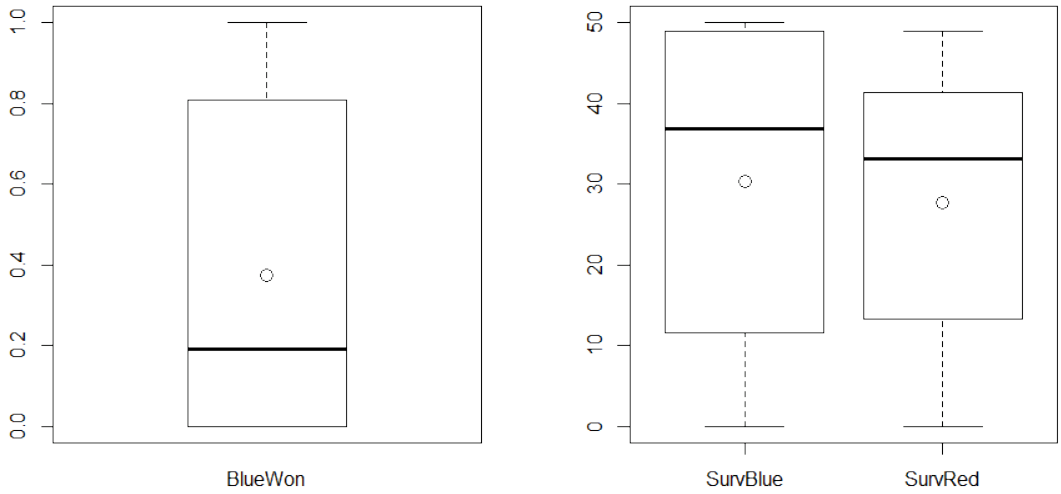


Figure 4.2: This figure shows the balanced design as all medians are centered except DistanceCenter and DistanceUAV. The vertical axes show the range of the variables while the thick horizontal line in each plot marks the median for this factor.

Next, we can compare the three response variables, the scale of which is shown in Figure 4.3. The mean of BlueWon is below 0.5, but this outcome is not unexpected, as we varied the initial position and CohesionMax of the blue swarm against the uniformly distributed Red swarm. However, we observe both extremes where, in addition to cases of never winning, the Blue swarm achieves 100% probability of success for certain design point(s). Considering the survival rate of both Blue and Red swarms (see Figure 4.3[b]), note that the numbers of surviving UAVs do not differ significantly between the swarms, where Blue has a higher mean but also a higher variance.

Let us explore the responses where Blue has at least 90% likelihood of success, that is, of destroying the Red's high value unit. Comparison of the distributions for this subset of data,



(a) (b)

Figure 4.3: Overview of data of the three responses. The circles are the means and the horizontal lines the median. Plot (a) shows a lower success rate for Blue as expected. Plot (b) expresses that both sides have approximately the same number of surviving UAVs even if Blue has a higher mean. But the variance is also higher.

see Figure 4.4, with the full response data shows significant differences for nearly all of the factors, with median factor values at either their high or low levels. The only exceptions are `DistanceCenter` and `DistanceUAV`, whose distributions are similar to those of the complete response data set. This suggests that perhaps initial positioning, represented by these two factors, does not affect the response, though further analysis is required to investigate if this is truly the case (e.g., due to interactions). All other variables indicate that there is an underlying, yet to-be-determined, influence on the results. We specifically investigate the exact behaviors in Section 4.1.4. It is also interesting to see that a high Blue success rate implies higher losses to the Blue UAV swarm, as seen by the low median value of `SurvBlue`.

4.1.2 Exploring Non-linearity in the Responses

For this portion of the analysis, we use an additive model where the usual regression coefficients are replaced by smooth functions, which offers more flexibility in fitting compared to linear models. The point we want to take advantage of is the ability to plot the smooth functions, which gives some idea of how the regression variables influence the response (see Faraway [32]). If we examine these plots, we should be able to find suitable transformations for the main effects.

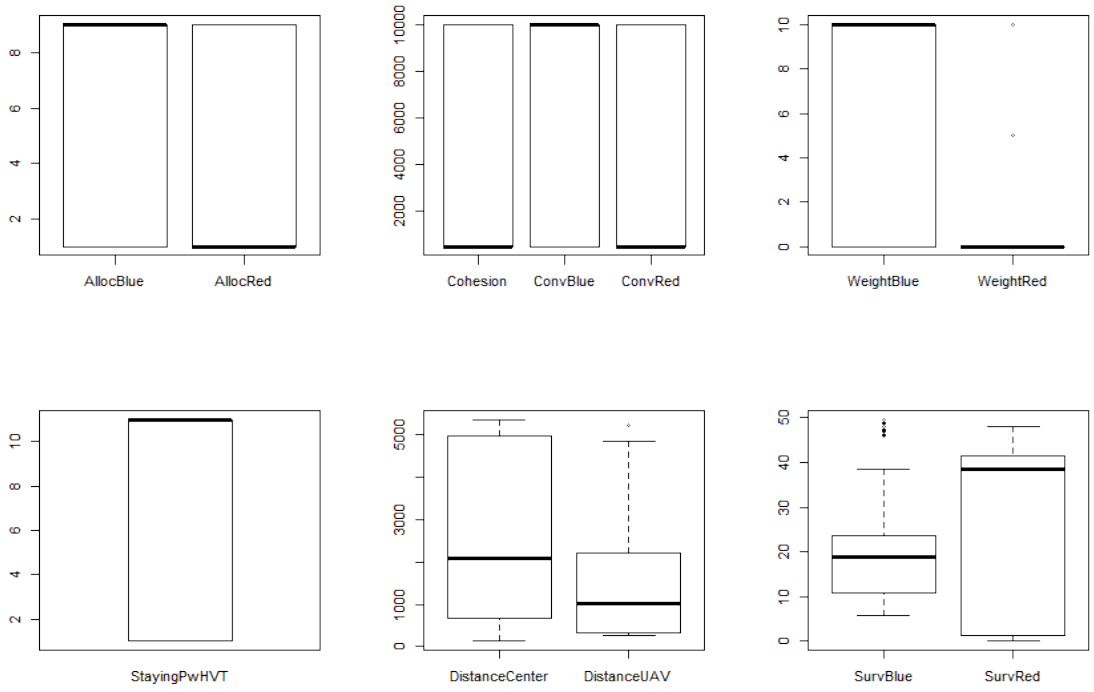


Figure 4.4: Boxplots for the subset of the design points and response data where Blue has a probability of success greater than 90%, i.e., `BlueWon` > 0.9. The vertical axes show the range of the variables while the thick horizontal line in each plot marks the median for this variable.

Let f_i for $i \in \{1,...,10\}$ be the smooth function for variable X_i, where X_1 corresponds to `NumAllocBlue` and X_{10} to `DistanceUAV` (as outlined in Section 3.1). Then (recalling the need to transform the response with the logit function):

$$\ln\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_{i=1}^{10} f_i(X_i) + \varepsilon$$

is the additive model we seek to fit. The transformation function determined for each of the ten factors is shown in Figure 4.5. As we can see, `NumAllocBlue`, `NumAllocRed`, `ConvergeRed`, `WeightBlue`, `WeightRed`, `DistanceCenter` and `DistanceUAV` are clearly nonlinear and therefore need some transformation prior to further analysis.

Figure 4.6 contains the same plot as Figure 4.5 but now includes the transformed effects (to be discussed shortly). All variables are much closer to linear as compared to before. The new model is of the form:

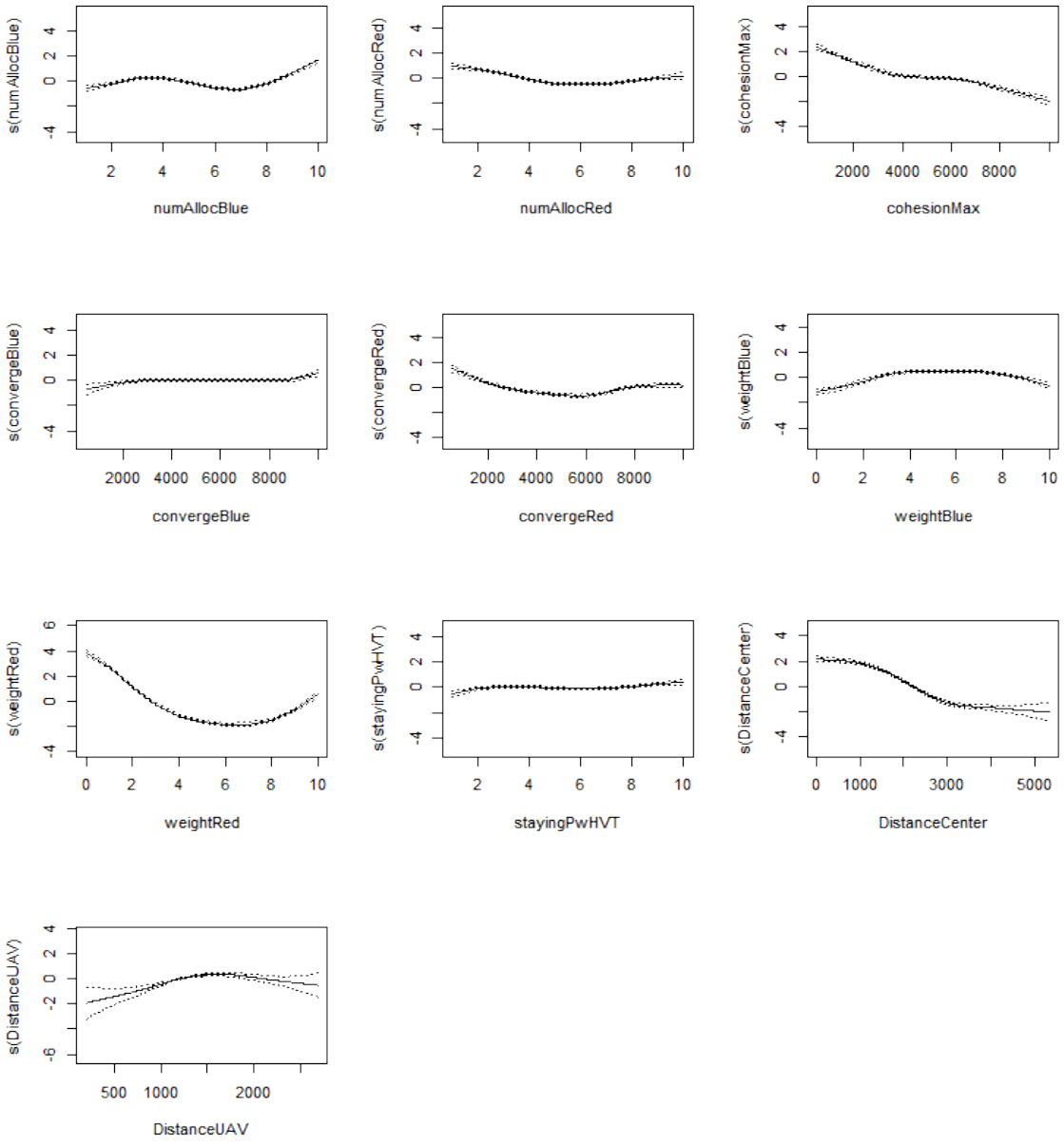


Figure 4.5: Smooth function for all regressors with 95% confidence interval. The plots indicate that `NumAllocBlue`, `NumAllocRed`, `ConvergeRed`, `WeightBlue`, `WeightRed`, `DistanceCenter` and `DistanceUAV` are nonlinear.

$$\ln\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_{i=1, i\neq 9}^{10} f_i(X_i) \tag{4.1}$$

$$+ \beta_{1_t}\sqrt{X_1} + f_{2_t}(X_2^2) + f_{5_t}(X_5^2) + f_{6_t}(X_6^2) + f_{7_t}(X_7^2) \tag{4.2}$$

$$+ \beta_{9l}X_{9_l} + \beta_{9_{tc}}X_{9_c} + \beta_{9_{tr}}X_{9_r} + \varepsilon \tag{4.3}$$

We can identify the appropriate transformations, with transformed functions and coefficients denoted by subscript "t," and begin by transforming `NumAllocBlue` with a square root term which corrects the sine wave-form that we see in Figure 4.5. The transformed term does not need a smoother in this case. For this first factor, the term is just modeled as in a regular linear regression, which we denote with the use of the regression parameter, β_{1_t}. Other parameters, namely `NumAllocRed`, `ConvergeRed`, `WeightBlue` and `WeightRed`, are fairly well transformed with quadratic expressions (Line 4.2).

However, we could not find a suitable transformation for X_9, or `DistanceCenter`. Another choice in such a case is to perform piecewise linear regression. The variable is divided into two or more intervals, and each part is treated as a linear regressor. The plot for `DistanceCenter` in Figure 4.5 indicates that there are three sections where each section could be modeled linearly, and the fitted lines would still be inside the confidence interval. Therefore, we have the left-hand side in the interval $[0, 1500]$, the center in $[1500, 300]$, and the right-hand side in $[3000, 5500]$. This modeling technique is explained in Faraway [32]. Finally, we get the following functional description, which replaces X_9 with three components:

$$X_{9_l} = \begin{cases} 1500 - X_9 & ; X_9 < 1500 \\ 0 & ; X_9 \geq 1500 \end{cases}$$

$$X_{9_c} = \begin{cases} 3000 - X_9 - 1500 & ; 1500 < X_9 < 3000 \\ 0 & ; X_9 \leq 1500 \text{ or } X_9 \geq 3000 \end{cases}$$

$$X_{9_r} = \begin{cases} X_9 - 3000 & ; X_9 > 3000 \\ 0 & ; X_9 \leq 3000 \end{cases}$$

With this piecewise linear representation, there is no need to further transform `DistanceUAV`, and we use the associated regression coefficients for the piecewise linear fits.

Having identified and addressed non-linearity issues in the main effects, in the next section we take Equation 4.1 as the basis for the regression and add interaction terms in order to analyze the simulation output.

4.1.3 Performing the Regression

With the transformation steps complete, we are prepared to identify important variables and interactions. The basic setup gives Equation 4.1, which enables the use of a linear predictor with the chosen transformations and, thus, satisfies the elements of a GLM. We can now use logistic

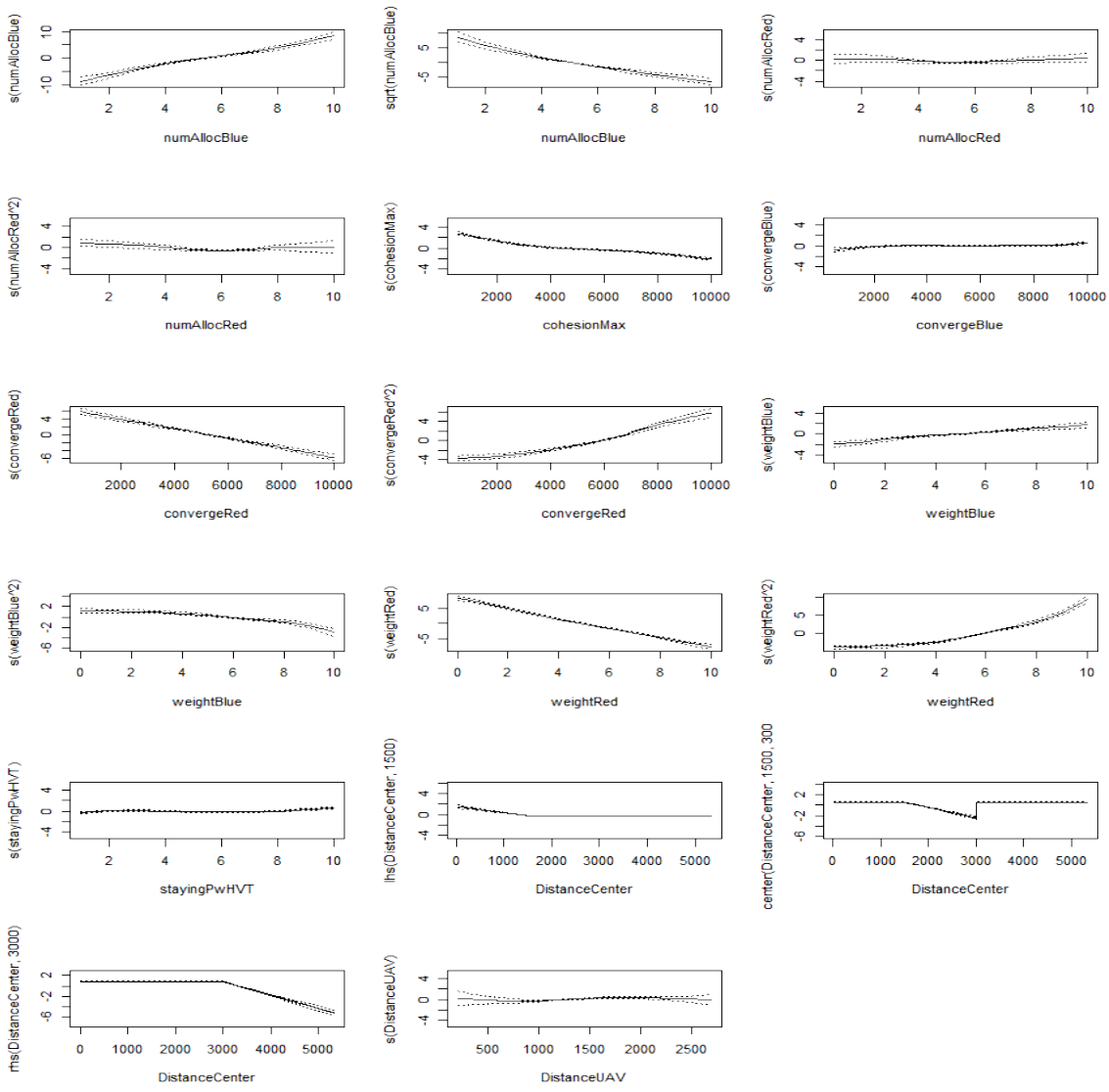


Figure 4.6: Smooth function for all main effects and transformed terms with 95% confidence interval.

regression to fit the model; however, we still do not know if a first-, second- or third-order model is sufficient in our case, that is, we do not know if two- or three-way interactions play a significant role. Recall that the CCD exposes interactions up to the third order only compounded by higher order terms. But do we really need these higher orders to capture insights from the model? To answer this question, we fit three models, namely a first-, second- and third-order model, and compare their predictive power. To ensure fairness of the comparisons, we anticipate having to normalize the regressors to the same scale to compare the chosen coefficients and to draw our conclusions before selecting one of these three.

Lasso Regularization

Prior to defining the various models, we begin by briefly explaining the proposed modeling approach. We know that logistic regression is the right choice for a binomial response. However, instead of applying conventional approaches, we explore an alternative technique of lasso regularization, which also provides a logistic model, but has at least three advantages in our case. First, lasso prevents over-fitting by restricting the complexity of the model, so we expect fewer terms. As we are interested in the most significant influences, identifying the simplest fit is advantageous. Second, the results of both approaches are comparable and the small difference can nominally be ignored with our purpose in mind. Third, the cross validation or variable selection part is computationally cheaper for lasso. In fact, we observe that it is nearly intractable to fit a third-order model in the normal way, while using the lasso approach takes only half an hour on an moderate computer. Lasso regression is further explained by Hastie, Tibshirani and Friedman [33].

The score function for a normal linear regression is the sum of squared errors

$$SSE = \sum_i (y_i - \hat{y}_i)^2$$

which is to be minimized. Instead, for lasso regularization, an additional penalty is included to control the complexity and to prevent over-fitting, as seen in Equation 4.4.

$$SSE + \lambda \sum_{j=1}^{k} |\hat{\beta}_j| \tag{4.4}$$

The βs are the regression coefficients and λ is the tuning or complexity parameter, and must be chosen for a specific model. A common way to determine the tuning parameter is via cross validation. Equation 4.4 represents the new score function, which is to be minimized, which drives coefficients to zero and leaves fewer terms remaining in the model.The following section describes the application of the lasso regularization approach to our specific dataset.

Model Comparison

For completeness, we conduct a comparison of the fitted models using both normal logistic regression and the lasso regularization approach, showing their similarity in results while the latter has computational advantages. Before describing the application of the lasso process in detail, we provide a summary table comparing the fitted models. Table 4.1 shows the predicted mean

error and its standard deviation for varying order models for both the normal logistic regression and lasso approaches. Note that in the case of the normal logistic regression, generation of a third-order model was untenable within reasonable computation.

order model	measures	normal LR	Lasso
first	mean	0.196	0.196
	SD	0.00245	0.00273
second	mean	0.136	0.138
	SD	0.00211	0.00229
third	mean		0.13
	SD		0.00223

Table 4.1: Comparison of different order models and regression techniques where mean it the error rate. The third-order model for normal logistic regression is intractable. But we assume that it would show almost the same figures as Lasso regression according to the results for the first and second-order model. Overall, the third-order model does not improve very much compared to the second-order model.

First of all, we can see that both approaches provide nearly equivalent results for the first- and second-order models. We would assume that this is also the case for the third-order model. More important to note, however, is that the prediction error for the third-order model decreases only marginally compared to the second-order errors. We determine that the third-order model provides little additional benefit but requires the addition of many more terms. This increase in the model's complexity is undesirable; thus, we conclude that a second-order model is suitable for the analysis of our data. Another point is that three-way interactions are hard to understand, to visualize, and to interpret, and we no longer have to address this issue.

As a note, the prediction power of the second-order model will be seen to be about 86%, which is very good for the simulation model. In fact, we expected a smaller value because warfare is always uncertain. Nevertheless, this result leaves sufficient ambiguity to satisfy our intuition and to convince us that we are on the right track. Table 4.1 summarizes the mean error rate for the different models. The error rate for the second-order Lasso model is 14%, which can be divided into type I errors, where the model incorrectly predicts a Blue success when it really is not, and type II errors, where the model predicts a Blue failure when it is a success. The type I error is about 7.2% and the type II error is about 6.6%. In order to identify whether the results indicate a bias for either error type, we compare both with a two-sided t-test, where the null hypothesis is that both are equal and the alternative is that they are different. The p-value for this test is 0.0501 with a 95% confidence level, which is (barely) larger than 0.05, and as such, we do not reject the null hypothesis. Therefore, we do not see bias in one direction or the other, which provides further confidence in the analysis of the simulation model presented below.

The Model

In this part we explain the modeling process in detail and describe how we arrive at the final list of terms that are important in representing swarm vs. swarm combat. We start with the simulation output for the CCD, which requires some pre-processing. Then we divide the data set into a training set, which we use for cross validation, and a test set, which is used to determine the prediction error. Afterwards, we can cross validate to choose the best λ for our model. This value also determines important main effects and interactions. Finally, we can compare the predicted values with the test set. The statistics tool, R, is our choice to do all these computation. The packages `gam` and `glmnet` are used in addition to the standard R packages.

Let us start with the data set obtained from the simulation (see Table 3.4). For the immediate analysis, we want to only explore `BlueWon` (Y_3). Recall that we replace `yAlpha`, `yBeta`, `zAlpha` and `zBeta` by `DistanceCenter` and `DistanceUAV`. The final data set, in which we have the desired one response and ten regressors is given in Table 4.2.

Y_3	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_{13}	X_{14}
1	1	1	500	500	500	0	0	1	799	5164
1	1	1	500	500	500	0	0	1	360	5239
0	1	1	500	500	500	0	0	1	415	5306
0	1	1	500	500	500	0	0	1	317	5296
0	1	1	500	500	500	0	0	1	1030	5212
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.2: Modified data set for analysis purposes where Y_3 is the column for `BlueWon` and X_{13} and X_{14} are `DistanceCenter` and `DistanceUAV`, respectively.

We get this data format by importing the CSV-files mentioned at the end of Section 3.4 into R and selecting the appropriate columns in the right order. Additionally, we transform `BlueWon` into a factor as this is needed for the follow-on analysis.

```
result = read.csv(".../resultDoE_CCD.csv", header = TRUE)
resultDoE = result[,c(15,1,2,3,4,5,6,7,8,16,17)]
resultDoE$BlueWon = as.factor(resultDoE$BlueWon)
is.factor(resultDoE$BlueWon)
```

It is common practice to divide the available data into two sets, with one training set to fit the model and one test set to determine how well the model predicts the unknown data. A rule-of-thumb suggests 10 to 25% of the whole data set should be used as the test set. We have 132,300 observations in total. We conclude that 10% of these data points are suitable in our case, and so we select randomly 13230 observations for the test set, denoted `data.test`, and have 119070

observations left in the training set, denoted `data.train`. The illustrative R code provides this partitioning.

```
ran = sample(132300, 13230)
data.test = resultDoE1[ran,]
data.train = resultDoE1[-ran,]
```

In the next step, we fit a saturated second-order model with normal logistic regression in order to generate all the interaction terms. The function `glm()` does this, as illustrated below:

```
resultDoE.glm.lasso = glm(
  BlueWon~
    (.
      -DistanceCenter
      + lhs(DistanceCenter, 1500) + center(DistanceCenter, 1500, 3000)
      + rhs(DistanceCenter, 3000)
    )^2
    + I(sqrt(numAllocBlue)) + I(numAllocRed^2) + I(convergeRed^2)
    + I(weightBlue^2) + I(weightRed^2)
    - lhs(DistanceCenter, 1500):center(DistanceCenter, 1500, 3000)
    - lhs(DistanceCenter, 1500):rhs(DistanceCenter, 3000)
    - center(DistanceCenter, 1500, 3000):rhs(DistanceCenter, 3000)
  family=binomial,
  data.train,
  x = TRUE
)
```

Recall we divided the interval for `DistanceCenter` into three parts to solve the non-linearity issue of this regressor. This fact explains why we take out `DistanceCenter` and replace it with the three parts `lhs`, `center` and `rhs`. We also add the quadratic terms. Finally, we remove the unnecessary interactions between the three parts of `DistanceCenter`.

The option `x = TRUE` ensures that the returned `glm` objects include a matrix with all terms and their values at each observation. This is a 119070×80 matrix in this case. We have 80 terms in the saturated model and want to choose the important ones.

```
x=data.matrix(resultDoE.glm.lasso$x[,-1])
```

Next, we want to compare the fitted $\hat{\beta}_j$s, but observe that the regressors are on different scales. For example, `NumAllocBlue` is defined in the interval $[1,9]$ while `ConvergeRed` is in the interval $[500,1000]$. Without addressing this issue, the resulting coefficients, $\hat{\beta}_j$s, would be different even if both have the same influence. We must first normalize the regressors to be on an equal scale, which can be performed using one of various methods. We choose one that

Montgomery and Runger [34] suggest, namely

$$z_i = \frac{X_i - \bar{X}_i}{s_{X_i}}$$

where $\bar{X}_i$ is the mean of variable X_i and s_{X_i} the standard deviation. If we do this normalization with all regressors (as follows), then all transformed factor value intervals have a mean of zero and a variance of one.

```
for(i in 1:(dim(x)[2])) {
    xMean = mean(x[,i])
    xSD = sd(x[,i])
    z[,i] = (x[,i] - xMean) / xSD
}
```

We need the response in a separate vector for the `glmnet()` function which runs the lasso regularization,

```
y=data.train$BlueWon
```

after which we can run `glmnet()` and plot the results.

```
resultDoE.lasso = glmnet(z,y,family = "binomial")
plot(resultDoE.lasso, xvar="lambda", label=T)
```

How the choice of λ influences the magnitude of the different $\hat{\beta}_j$s and the number of terms left in the model is shown in Figure 4.7. The goal is to determine a λ that minimizes the score function, that is, Equation 4.4. Function `cv.glmnet()` does this for us using ten-fold cross-validation, illustrated and described below (see [33]).

```
resultDoE.cv.lasso = cv.glmnet(z, y, family = "binomial", maxit = 500000)
plot(resultDoE1.cv.lasso)
```

The use of ten-fold cross-validation means that the data set (here denoted z) is divided randomly into ten partitions, and one of these partitions is always the validation set while the others are used to fit the model. In other words, the function fits ten models for every λ where every partition is used as a validation set once. Then, the mean deviation and its variance is computed over the ten replications per λ. The results are shown in Figure 4.8. The deviation is smallest for 78 terms present in the model. However, since all the deviations are only estimates and we want the minimally complex model, such that `cv.glmnet()` uses the ”one-standard-error“ rule where a λ is chosen which is one standard error to the right of the minimum. Faraway [32] also describes this method.

```
lambda.lasso = resultDoE.cv.lasso$lambda.1se
```

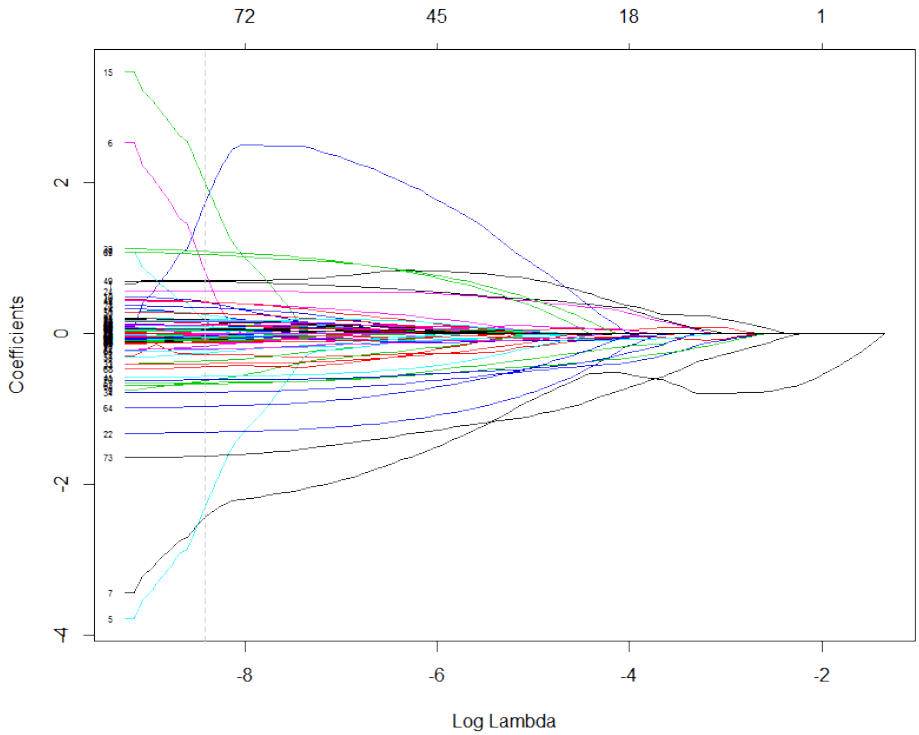


Figure 4.7: Lasso Regularization: Horizontal axis shows the log of the tuning parameter, λ (bottom axis) which is equivalent to the number of terms left in the model (top axis). Vertical axis determines the size of the coefficients.

The choice of $\lambda = 0.00022$ is shown in Figure 4.8 with a dotted line slightly to the right from the minimum in the plot. Figure 4.7 contains a corresponding dotted line which marks the chosen λ-value. Finally, we are left with 74 main effects and interactions in the model.

The next step is to determine the prediction power of the model. We use the earlier separated test set for this purpose. Again, we must fit a saturated model with the `glm()` function first to get the `x.test` matrix and to standardize the variables. We only have to ensure that we use the means and standard deviations we obtained earlier from the training set. Then we predict the $\hat{y}$s for the test set. All of these values are in the interval $[0, 1]$. Recall that the response variable, `BlueWon`, is binary, that is, either 0 or 1. The inline R function `result` below is used to conclude 0 if $\hat{y}$ is below 0.5 and 1 if $\hat{y}$ is above 0.5.

```
y.hat.lasso = predict(
                resultDoE.lasso,
                type = "response",
                newx=z.test,
                s=lambda.lasso
```

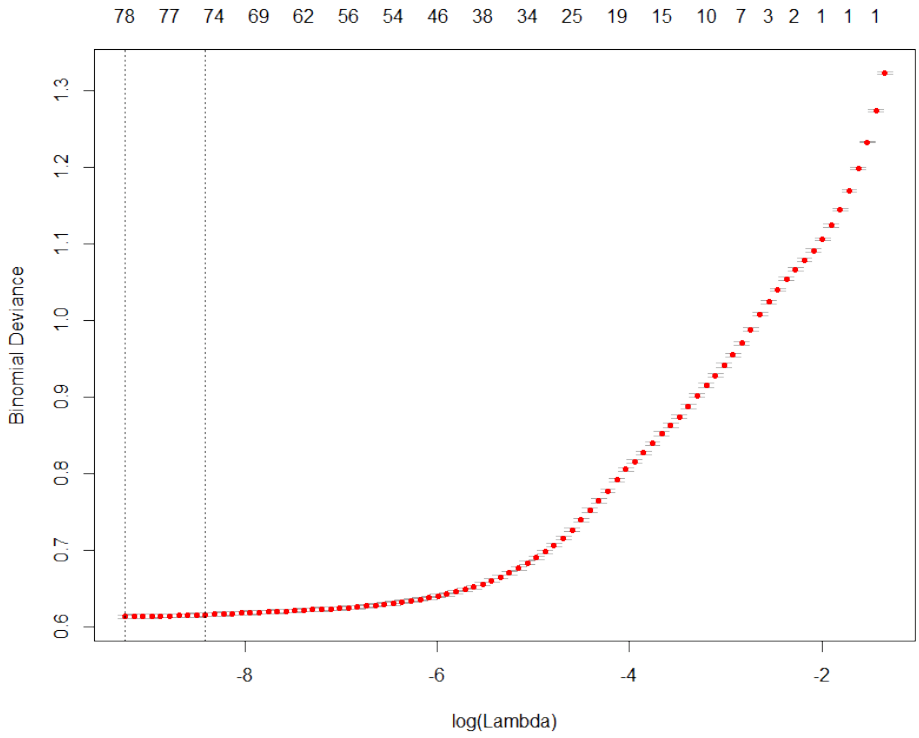


Figure 4.8: Lasso Regularization: Cross-validated λs against deviations.

```
            )
result = function(x) if(x >= 0.5) 1 else 0
y.hat.lasso = apply(y.hat.lasso, 1, result)
error.lasso = y.test - y.hat.lasso
mean(error.lasso^2)
sd(error.lasso)/sqrt(13230)
```

If we subtract the predicted values from the values in the test set and compute the mean over the resulting value, then we get the prediction error, for which the values are either 0 or 1. If the model predicts well, then `error.lasso` is 0. If it predicts poorly, then `error.lasso` results in either 1 or -1. The square of the errors gives just the absolute value and the mean is computed over the positive errors. The results of this analysis provide the values summarized in Table 4.1.

The `resultDoE.lasso` object comprises a matrix `beta` that includes the $\hat{\beta}$s for every tested λ. We pull out the $\hat{\beta}$s for our choice of lambda. Appendix B provides the complete listing of all terms that are included in the final model and their corresponding $\hat{\beta}$ values.

```
resultDoE1.lasso.coef = resultDoE.lasso$
            beta[,which(resultDoE.lasso$lambda == lambda.lasso)]
```

```
betas = as.matrix(resultDoE.lasso.coef[resultDoE.lasso.coef != 0])
```

The final model comprises 75 terms, indicating that only five terms of the saturated second-order model are not important to explain the response. This number of significant factors is quite a lot, as we expected to see fewer terms in the model, that is, only a handful of significant factors. We also see that, even having standardized the variables, the range of coefficient values is still large (as seen in Appendix B). We concentrate further analysis on those terms that are deemed more important or relevant to the scenario, rather than trying to explain the sensitivity for all factors. As a method for drawing this line, we decide to take the mean of the $\hat{\beta}$s and explore those terms with a $\hat{\beta}$ larger than the mean. The resulting set of terms we discuss in the next section, are shown in Table 4.3.

Term	$\hat{\beta}$
weightRed	-2.40700031
convergeRed	-2.34268830
I(convergeRed^2)	2.02622621
I(weightBlue^2)	1.75709594
weightRed:rhs(DistanceCenter, 3000)	-1.63271293
numAllocBlue:weightBlue	-1.30740299
numAllocRed:weightRed	1.08292601
weightRed:stayingPwHVT	1.04541938
weightBlue:stayingPwHVT	-0.97929942
weightBlue	0.82523436
numAllocRed:stayingPwHVT	-0.79081038
weightBlue:weightRed	-0.70730609
convergeBlue:weightBlue	0.69240074
numAllocBlue	0.68324259
convergeRed:weightRed	-0.66977778
DistanceUAV	-0.65948801
weightRed:DistanceUAV	-0.61042046
cohesionMax:weightBlue	-0.57027147
numAllocBlue:stayingPwHVT	0.55291950
weightBlue:rhs(DistanceCenter, 3000)	-0.47040495
cohesionMax:DistanceUAV	0.40974145
numAllocRed:weightBlue	-0.40864333
cohesionMax:weightRed	0.39484181

Table 4.3: Most influential main effects and interactions in the final model and their coefficient values.

4.1.4 Results

Having performed the above rigorous statistical analysis to provide a regression model that describes the response, we can begin to draw conclusions based on the fitted coefficients. It seems the idea of variable selection does not work in this case because even if not all main effects appear as important, they are at least represented in the interaction terms. Therefore, we cannot conclude to reject any of the used variables in the simulation. Next, we want to discuss the terms that have the highest significance according to the magnitude of their fitted coefficient.

`WeightRed` and `ConvergeRed` have the largest (absolute) $\hat{\beta}$-values, with values below -2, signifying that these two are the most important factors in this model. From Blue swarm's perspective, this fact is disheartening as both variables are controlled by the Red swarm's tactical choice. For example, we see that $\hat{\beta}_7 = -2.41$, which means that a higher weight factor chosen by the Red swarm (i.e., Red swarm agents are more offensively driven than not) decreases the Blue swarm's effectiveness and likelihood of destroying the Red's high value unit. In fact, if we delve further by examining the interaction plot of the weight factors for the Blue and Red swarms, respectively, shown in Figure 4.9, then we observe that Red must simply maintain a high weight factor, and Blue's choice no longer has any influence on the result. Recalling that `BlueWon` is Blue swarm's success rate, we see that Red simply must choose a weight factor of five or above to decrease Blue's success dramatically.

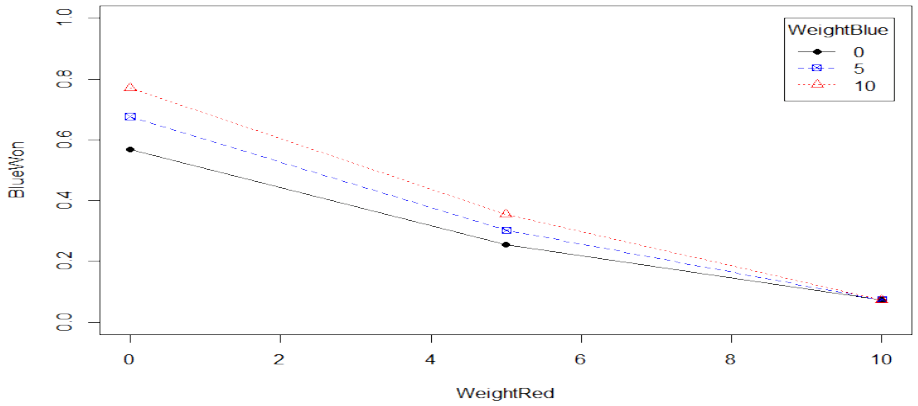


Figure 4.9: Interaction of the weight factor for the Blue and Red swarms. This plot says that as long Red maintains a high weight factor it is able to decrease Blue's probability significantly. Blue's choice does not matter in this case.

Another perspective is to look at the number of surviving UAV on both sides. Figure 4.10 shows these values, namely `SurvBlue` and `SurvRed`, as a function of the Red swarm's choice

of weight factor. The plot shows that the Blue swarm dominates with respect to its surviving force ratio as Red uses a higher weight factor. Though it might lose its high value unit first to the aggressively offensive Red swarm, the Blue swarm retains much of its UAV fleet and would nominally be able to counter-attack and destroy the Red's high value unit as well.

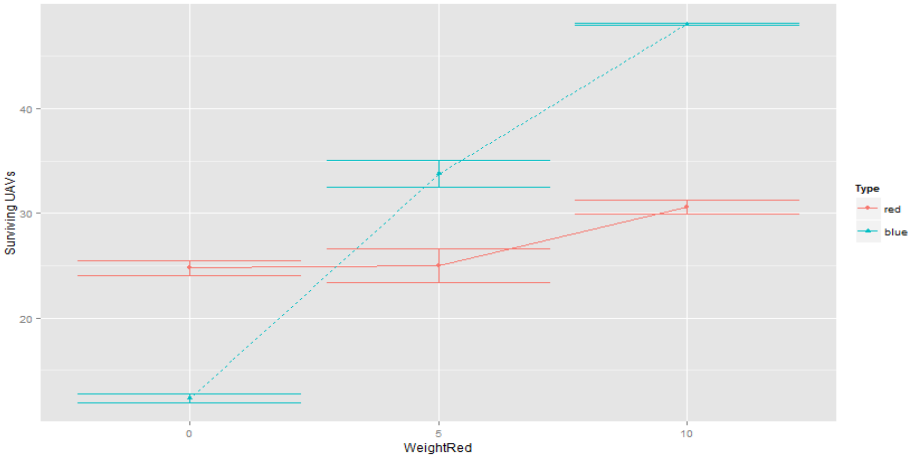


Figure 4.10: Surviving Blue and Red UAVs at the end of the engagement for different red weight factors.

This disparity between the success measure (i.e., `BlueWon`) and the surviving UAVs measures (i.e., `SurvBlue`, `SurvRed`) highlights a current limitation of the agent-based simulation model in its implementation. Rather than terminating the simulation after the first HVT is destroyed, the simulated scenario should continue to execute until both HVTs are destroyed (or until one side is completely dominated, i.e., all of its UAVs and its HVT are destroyed). Such a modified termination criterion would lead to draws as well, and as such, different priorities or weightings for successes and draws would change the analysis and the findings presented in this work. The extension to this enlarged scope of the scenario is left for follow-on studies.

Continuing with the analysis, recall that we transformed `ConvergeRed` with a quadratic effect in order to solve non-linearity in the main effect. Both linear and transformed terms associated with `ConvergeRed` appear as significant factors in the regression, and merit further exploration of the impact. `ConvergeRed` and response variable `BlueWon` are shown in Figure 4.11. Recall that `ConvergeRed` defines the distance to the Blue HVT at which a Red UAV starts to fly directly towards it. In other words, until the range is less than the `ConvergeRed` threshold distance, the UAV flies roughly towards the Blue half but not directly towards the HVT. Therefore, a higher value means that the swarm agents begin to converge and target the enemy HVT earlier (i.e., at greater range). The plot indicates that maintaining a more dispersed formation is more beneficial

for Red than converging too early in the transit; however, this effect turns around at some point, where at approximately midfield, (approximately at a distance of 4750 m from the Blue HVT), there is an inflection point where concentrating forces initially provides a tactical advantage. The design implemented in this study does not provide sufficient resolution to determine the exact point, but can be conducted in near-term future research.

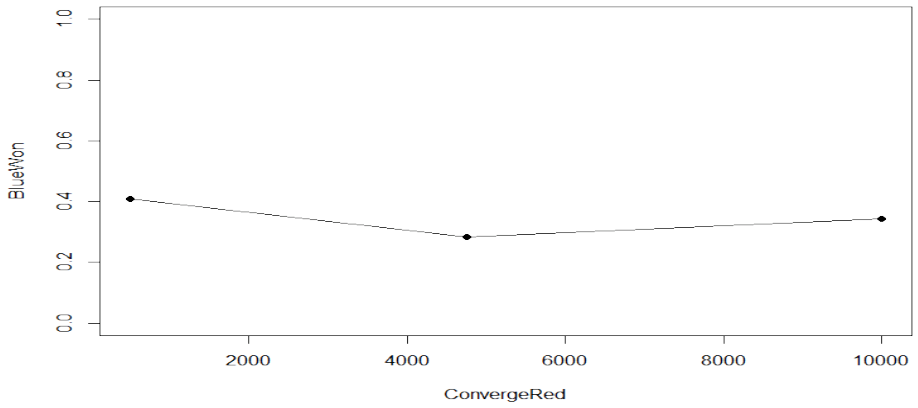


Figure 4.11: Influence of `ConvergeRed` and the quadratic transformation on the response. There is a saddle point somewhere around 5000 meters. The swarm should should wait with the final attack on the HVT, but also not too long.

Despite already knowing that Blue's choice of `WeightBlue` has little influence on the outcome *if* Red chooses a high weight factor, we still wish to explore this effect as it appears important according to the magnitude of the regression coefficient, with an emphasis on the quadratic nature of the dependency. Indeed, if we look at Figure 4.12, we see a bigger increase in the success rate for higher weight factors. Accordingly, Blue should maintain a high weight factor in general as this tactical approach is more beneficial, noting that the present simulation study emphasizes first kill of the opponent's HVT (rather than including draw conditions).

Another interesting analysis point appears in exploring the impact of the initial positioning of the swarm agents. The coefficient for the interaction between `WeightRed` and `DistanceCenter` signifies its importance. The different slopes for different weight factors are shown in Figure 4.13. More important, however, is the fact that we see a decreasing success rate with increasing distance from the center; that is, increasing `DistanceCenter`. `DistanceUAV` is the other factor that describes the initial positioning. The coefficient is negative and suggests that a more compact formation is more beneficial. Therefore, the best Blue can do is a centered and concentrated initial configuration. This tactical choice makes sense as `WeightRed` is high

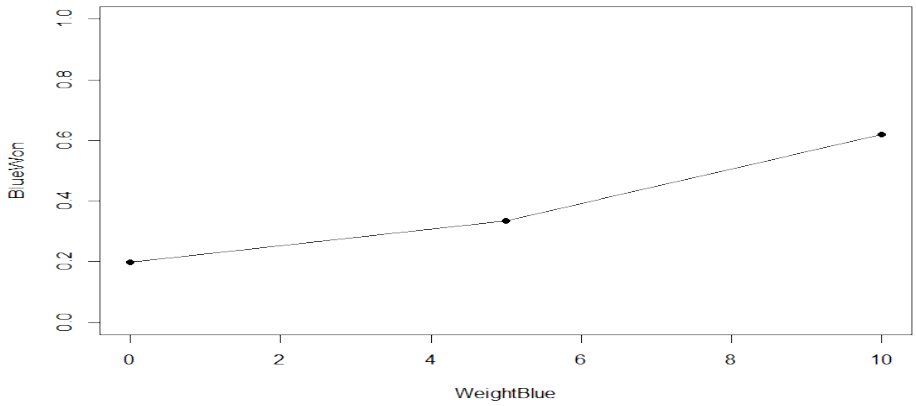


Figure 4.12: Influence of `WeightBlue` and the quadratic transformation on the probability of Blue success.

and the saddle point for `ConvergeRed` is somewhere near the middle of the battle space. Blue should expect most of Red's forces in the center, such that centering its own force works to concentrate firepower to fight against the inbound threat.

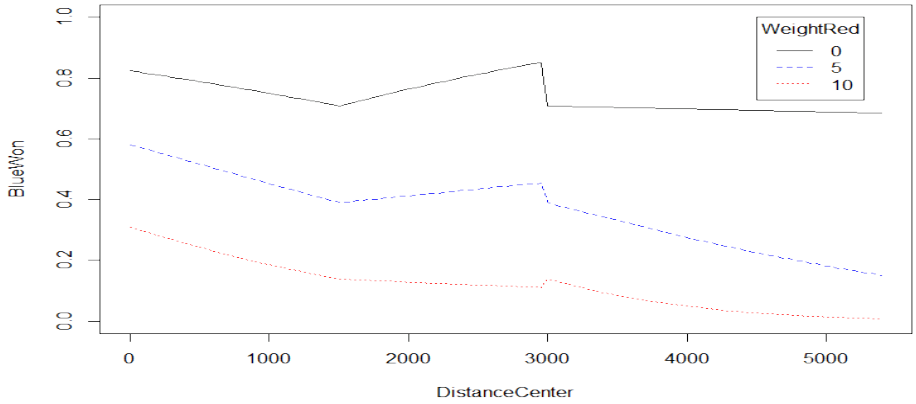


Figure 4.13: Interaction between `DistanceCenter` and `WeightRed`. The Blue swarm should maintain a more centered position in order to increase success.

If we now inspect Figure 4.14, we see an unexpected interaction between `NumAllocBlue` and `WeightBlue`. It makes sense that with a weight of zero (i.e., UAVs are completely defensive), a higher number of allowed allocations of defending UAVs to attacking UAVs increases the probability of success. However, for a maximum value for the weight factor of 10, we would have expected to see a leveling out of the response as the Blue UAVs would concentrate only on

the Red HVT and not aim to defend against Red UAVs. This behavior does not seem to be true. Having already stated that Blue should maintain a high weight factor, `NumAllocBlue` should nominally be kept low. Hence, an intermediate value for the weight factor between the low and the high levels does not benefit the Blue swarm's effectiveness. Rather, `WeightBlue` should either be at high or low levels, with `NumAllocBlue` chosen accordingly. This insight implies that a *homogeneous* UAV swarm fleet should be designed for pure offense or pure defense, with little benefit of a mixed strategy. However, future research could investigate *heterogeneous* swarms, either by platform or by task assignment, where some agents are designated for the offensive strike mission and others are specifically reserved for defensive countermeasures to attacking UAVs.

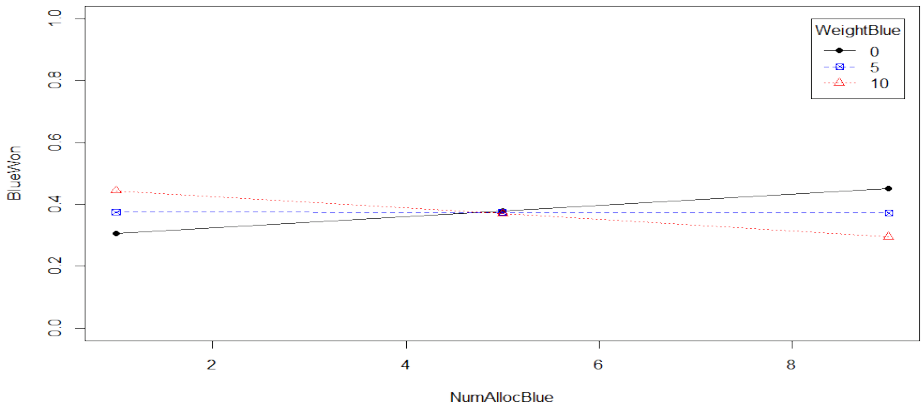


Figure 4.14: Interaction between `NumAllocBlue` and `WeightBlue`. A high Blue weight factor demands a low number of allocations and vice versa.

Further inspection of the relationship between `NumAllocRed` and `WeightRed` yields a similar relationship, shown in Figure 4.15, as seen previously; that is, Red should also choose a low number of allocations in order to support a high weight factor.

The next two plots in Figure 4.16 also illuminate an interesting behavior. A low staying power of the HVTs (assumed the same for both sides) favors a high Blue weight factor, and conversely, a high staying power implies using a low Blue weight factor. One reasonable explanation could be that the losses experienced with a high weight factor are too great to retain sufficient forces to achieve the main objective. We see an analogous behavior for Red (noting that red's objective is to minimize `BlueWon`). Therefore, according to the presented analysis, the assumed staying power of the respective high value units should appropriately inform the choice of the weight factor for at least the blue swarm.

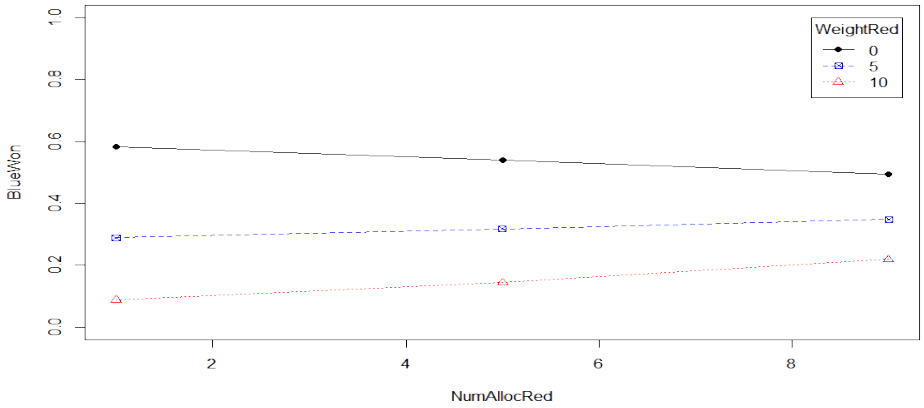


Figure 4.15: Interaction between `NumAllocRed` and `WeightRed`. Again, the number of allocations should be high if the weight factor is low and contrariwise.

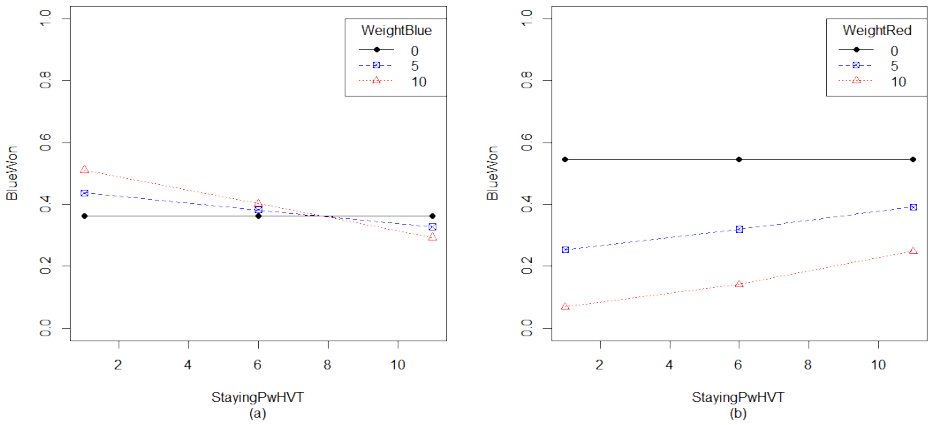


Figure 4.16: Interaction between `StayingPwHVT` and `WeightBlue` (a)/ `WeightRed` (b). `StayingPwHVT` cannot be controlled by the swarm and is therefore the basis for the choice of the weight factor. Blue should have a high weight factor for a low staying power and vice versa. Red does best with a high weight factor every time.

A final conclusion that merits mentioning is that `CohesionMax` turned out to not carry as much importance as we might have expected, as more coordination in swarm motion may have assisted in keeping forces concentrated. In fact, the coefficient is negative and suggests that strong cohesion of the swarm does not provide a benefit. This result also speaks to a broader question of the value of coordination among distributed agents, and perhaps the benefit or penalty therein. Recall, however, that cohesion is only one of three parts that define the swarming behaviors implemented in the simulation model, among shared broadcast communication of detected targets

and local negotiation of target allocations. We have not investigated these capabilities with the current simulation study because both swarms are treated as possessing identical coordination capabilities. A richer future analysis that explores the advantages or disadvantages of varying coordination levels as well as disparities in their implementation between Blue and Red swarms would likely be an interesting and insightful avenue of research.

4.1.5 Sensitivity Analysis

The presented results in the previous section provide general insights on how values should be chosen for the defined parameters in our scenario to maximize the success of either side. In this section, we consider specific recommendations and construct a new design point reflecting those insights, as described in Table 4.4.

Variable Name	Value
NumAllocBlue	9
NumAllocRed	9
CohesionMax	500
ConvergeBlue	10000
ConvergeRed	5000
WeightBlue	10
WeightRed	10
StayingPwHVT	6
YAlpha	1001
YBeta	1001
ZAlpha	1001
ZBeta	1001

Table 4.4: Parameter setting for additional analysis.

Given that the staying power of an HVT is a function of the asset itself and is considered outside the influence of the swarm parameters, we choose a nominal value of six. All other parameters are dictated by the swarm, and guided by the insights obtained earlier. Recall that the weight factor is the most important parameter for both Red and Blue swarms. We therefore choose a value for this variable first. As it interacts with staying power, we set `WeightBlue` and `WeightRed` to ten for both swarms. The choice for the number of allocations also depends on the weight factor, and based on Figures 4.14 and 4.15, this parameter should be set to one for both Red and Blue. We also concluded that cohesion should be as low as possible, for which a defined range of 500 meters is an appropriate value. We set `ConvergeRed` to a value of 5000 because we have the quadratic influence that exhibited an inflection point approximately near this value. On the other hand, `ConvergeBlue` does not exhibit a nonlinear effect, and

its regression coefficient (refer to Appendix B) is positive. Hence, the parameter should be as large as possible, with 10000 meters as the maximal range. Finally, we configure the values for specifying the initial positioning. The analysis in Section 4.1.4 suggests a centered and closed setup, which we obtain if both distributions are symmetric and have a low variance. This is the case if `Alpha` and `Beta` have approximately the same value and are as large as possible, namely a value of 1001. Note that the Red forces are still uniformly distributed in the pre-positioning bin.

We run this design point with 1000 replications and conduct a two-sided t-test on the `BlueWon` measure, where the null hypothesis is that Red and Blue have an equal chance of success and the alternative is that the probabilities to succeed are different.

$$H_0 : p_{blue} = p_{red}$$
$$H_a : p_{blue} \neq p_{red}$$

Recall that the MOE is defined to be one for a Blue success in a given single run and zero for a Red success. Therefore

$$\bar{p}_{blue} = \frac{\sum BlueWon}{1000}$$
$$\bar{p}_{red} = 1 - \bar{p}_{blue}$$

are the mean probabilities for a Red and Blue success.

The t-test shows that $\bar{p}_{blue} = 0.532$. The p-value is 0.043 and therefore below 0.05. Hence, we reject H_0 and H_a is true, performed and output in R as follows.

```
> t.test(resultDoE.data$BlueWon, alternative = "two.sided", mu = 0.5)
  One Sample t-test
  data:  resultDoE.data$BlueWon
  t = 2.027, df = 999, p-value = 0.04293
  alternative hypothesis: true mean is not equal to 0.5
  95 percent confidence interval:
  0.5010208 0.5629792
  sample estimates:
  mean of x
  0.532
```

For the selected design of the swarm systems, Blue has the advantage in this particular case. The main disparity arises due to the initial positioning of both swarms, as the Blue swarm specifically selected its initial spatial configuration, which is superior to the Red's uniformly distributed positioning.

4.2 Markov Process Analysis

With the in-depth simulation analysis performed, this section investigates insights generated by the analytic formulation of the swarm-versus-swarm engagement using a Markov process model introduced in Section 2.3. By designing and assigning values to the defined probabilities and running the formulated computations, we can analyze the steady state properties of the model, followed by further study of its sensitivity to these transition probabilities.

4.2.1 End State Analysis

We begin this analysis by summarizing the appropriate transition probabilities and their assigned values in Table 4.5 (with discussion of the design choices following), which recalls the definitions presented in Table 2.3.1. Recall by construction that definition and design of these probabilities is sufficient for both the one-on-one and swarm-versus-swarm engagements, since the latter leverages the estimates constructed in the former.

Probability	Value
$p_{0,HR} = p_{0,HB} = p_{B,HR} = p_{R,HB} = p_{B,HB} = p_{RHR}$	0.01
$p_{0,B} = p_{0,R}$	0.49
$p_B = p_R$	0.60
$p_{B,R} = p_{R,B}$	0.10
$p_{B,0} = p_{R,0}$	0.28

Table 4.5: Transition probabilities for the one-on-one combat (first level) Markov process.

The values chosen for this first level Markov analysis are presented as nominal, representative values. We define the baseline case where both sides are assumed to be balanced, in both force strength (UAV swarm size) and swarm capabilities (identical forces). The transitions to the states, `HVTR` and `HVTB`, should be rare events because the UAVs are required to transit to the HVT and will most likely encounter opposing UAVs which force them into dogfights. We specify this notional value as 0.01. We do not differentiate between the different source states because we do not have any evidence that there are differences.

The model starts always in state `0`, the unengaged state. The probability for either side to reach a favoring state should be 0.5 in order to reflect equal forces. The next possible states, starting in state 0, are $HVTR$, BvR, $HVTB$ and RvB where $1 = p_{0,HR} + p_{0,B} + p_{0,HB} + p_{0,R}$. Two of the states are preferred by Blue and two by Red. Therefore

$$p_{0,HR} + p_{0,B} = 0.5$$
$$p_{0,HB} + p_{0,R} = 0.5$$

and it follows

$$p_{0,B} = 0.5 - p_{0,HR} = 0.49$$
$$p_{0,R} = 0.5 - p_{0,HB} = 0.49$$

as we already fixed $p_{0,HR}$ and $p_{0,HB}$.

The probabilities, p_B and p_R, describing the successful defeat of an opposing UAV are similarly assumed to be equal, since both have an equal chance to get into a firing position and shoot the opposing UAV down. We set this value to 0.6 to reflect that, given identical platform types, it is more likely that the evader is taken out of action than it is to manage to evade or to reverse roles and become the pursuer itself. We assume that it is harder for an evader to become a pursuer (again, due to the equivalent platforms) than to escape the pursuit, for which we set $p_{B,0}$ and $p_{R,0}$ to 0.28 and $p_{B,R}$ and $p_{R,B}$ to 0.1.

Now we can use Table 2.3.1 and Equation 2.1 to compute the steady state probabilities, presented in Table 4.6. Each instance of combat always starts in state `0`. Therefore we are only interested in the first row of the table which comprises the end state transition probabilities from state `0` to all four end states.

	B	R	HVTR	HVTB
0	0.47	0.47	0.03	0.03
BvR	0.75	0.21	0.02	0.02
RvB	0.21	0.75	0.02	0.02

Table 4.6: Steady state probability matrix for one-on-one combat (level1).

As expected due to the symmetry of construction of this baseline case, both Red and Blue UAVs have an equal chance to down the opposing UAV and the odds of individually destroying the enemy HVT are low.

We take these four steady-state probabilities and use them for the second level as formulated in Section 2.3.2. We again apply Equation 2.1 to the swarm-versus-swarm transition probability matrix presented in Table 2.3.2, and compute the steady state probabilities for the multi-UAV

engagement. Rather than showing matrix B^2 in its entirety, we present Figure 4.17 which illustrates the steady state probabilities graphically. Recall that in this swarm-versus-swarm case, the model always starts in the state `50/50`, allowing us to only consider the probabilities from this initial state to one of the end states.

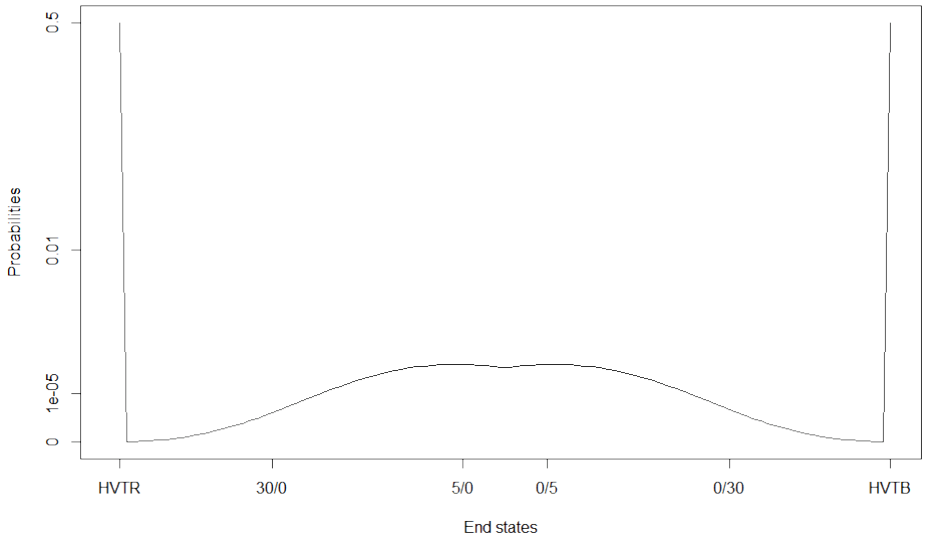


Figure 4.17: Steady state probabilities for multi-UAV engagement. A destroyed HVT is much more likely than a completely defeated swarm.

We can see that a destroyed HVT is most likely, while the probability of having one side completely eliminate the opposing swarm is almost zero. This fact can be easily explained, as the Markov process has the same issues as the simulation model discussed previously, in which the process is assumed to terminate (by design of the absorbing states) as soon as an HVT is taken out of action. The number of surviving UAVs on both sides does not play a role in either the previously or presently described models. Though such extensions can be implemented in future studies to generate a more realistic operational scenario (in which the fight does not necessarily end as modeled), the fact that both the simulation and the theoretical models show analogous behaviors helps convince us that the studies are consistent.

4.2.2 Sensitivity Analysis

The baseline model above assumes both sides are totally balanced, but fails to provide interesting insights that could be used as decision recommendations. We wish to explore variations of this model and their behavior in greater detail in this section. In order to do so, we change one

or two of the transition probabilities in level one (individual UAV engagements) and observe its impacts on the swarm versus swarm context.

Results of these variations, such as shown in Figure 4.17, provide a starting point. Note that the y-axis is not linear, to show that there is at least a small chance that a swarm is totally defeated and `HVTR` and `HVTB` are not the only end states with steady state probabilities greater than zero, albeit nearly so. For this reason of scale and for purposes of identifying general insights, we use a scenario with 10 versus 10 UAVs in this section. As such, the probability values are less extreme while still enabling easier exploration and understanding from the model while telling the same story as the envisioned 50 versus 50 UAV swarm model.

First, we want to explore the influence of the HVT end states. We already saw that both HVT end states have a high steady state probability. How does this change if we decrease or increase the transition probabilities? An increase in probability in a given transition would force that transition to the end states more likely, or alternatively a decrease in the probability to switch to other states, such as a decrease in the likelihood to get into a dogfight and perhaps correspondingly, reflect a more offensive attack preference. We discussed such a behavior previously as we identified the weight factors in the simulation model as the driver of offensive versus defensive agent-based behaviors. These transition probabilities of engaging the HVT provides this connection between the two modeling formulations. For further study, we fix Red's transition probabilities and change only Blue's values, $p_{0,HR}$, and tying other parameters accordingly, i.e., $p_{0,HR} = p_{B,HR} = p_{R,HR}$, as these parameters relate to the offensive attack preference of the Blue against the Red HVT. The probabilities vary from 0.0001 to 0.02, and we observe results where $p_{0,HR} \leq p_{0,HB} = 0.01$ and $p_{0,HR} \geq p_{0,HB} = 0.01$. Figure 4.18 indicates these state transitions are a dominant factor in the model, as it can be seen that Red has the advantage (i.e., the Blue HVT is most likely destroyed) for low values of $p_{0,HR}$. However, as $p_{0,HR}$ increases, the likelihood of arriving at the end state where the Red HVT is destroyed increases quickly, noting that all other steady state probabilities are decreasing as well.

Important to note is the similarity in Figure 4.18 as was observed in Figure 4.17, but on a different scale. This straightforward ability to scale the probabilities according to the swarm sizes provides flexibility in our analysis, allowing easier understanding of the sensitivities regardless of whether studying the 10 versus 10 or the 50 versus 50 UAV scenarios.

As another area to explore, what if we do not assume identical UAVs for both sides, such that one side has a tactical advantage over the other in one or more characteristics? The value of

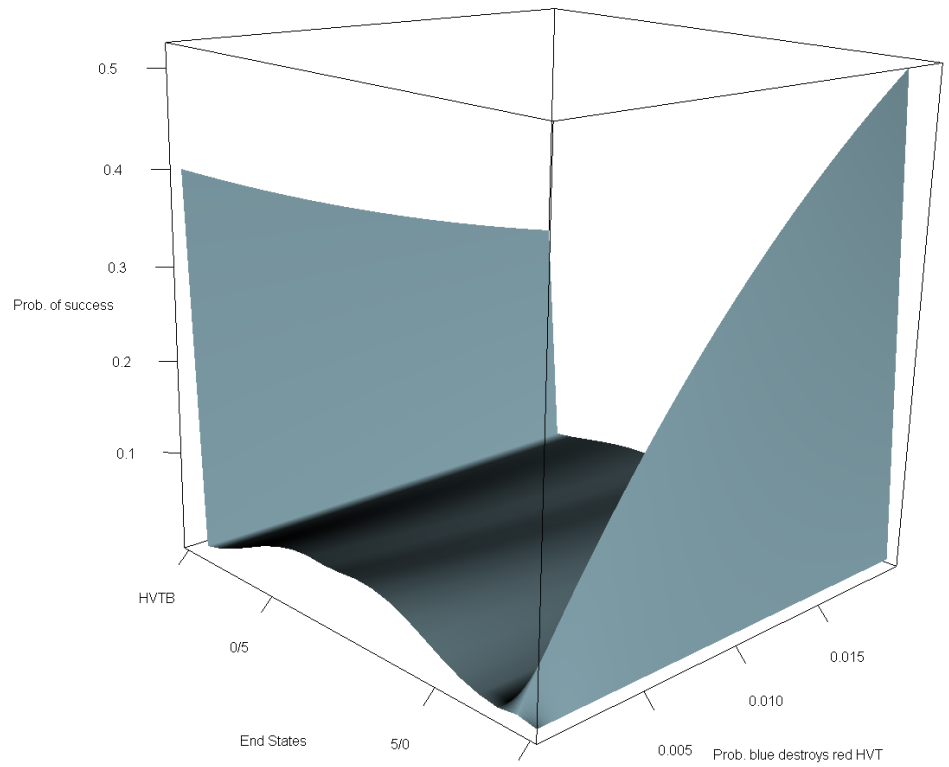


Figure 4.18: Steady state probabilities for different values of $p_{0,HR}$ (or equivalently, $p_{B,HR}$ and $p_{R,HR}$). The magnitude of this probabilities refer to the weight factor in the simulation model. The plot shows that an offensive swarm is superior to a more defensive one.

this sensitivity analysis is to show how much tactical advantage is necessary to achieve a desired specified success rate, which then has impacts on system-level design and development of UAV platforms. In order to explore this space, we vary $p_{0,B}$ and $p_{0,R}$, which reflect the probabilities of the Blue or Red UAV, respectively, entering an engagement with the edge. In other words, $p_{0,B}$ gives the probability that a Blue and a Red UAV enter into a dogfight while both are previously unengaged, and that Blue has the advantage. Recall that all outgoing edges of a given state must sum to one, such that if we increase $p_{0,B}$, for example, we necessarily must decrease $p_{0,R}$. For the presented example, consider $p_{0,B}$ to range from 0.09 to 0.89, with changes to $p_{0,R}$ accordingly. Values $p_{0,B} > p_{0,R}$ imply that the Blue UAVs have better capabilities (e.g., enhanced speed, sensors, swarm performance) or better tactical position (e.g., higher flight ceiling). These kinds of behavior also influence the states `BvR` and `RvB`. For example, assume $p_{0,B} + p_{0,HR} = 0.6$ and consequently $p_{0,R} + p_{0,HB} = 0.4$, and Blue pursues Red (i.e., state `BvR`). Then we would expect that p_B increases and p_{BR} and p_{B0} both decrease due to the better performance of Blue. That is, Red has less opportunities to decide the outcome of

the engagement. Conversely, these correlations also hold for `RvB`. The mentioned probabilities are changed proportionally to the change in $p_{0,B} + p_{0,HR}$. In our example, $p_{0,B} + p_{0,HR} = 0.6$ and therefore there is a change of 10% compared to the baseline model in Section 4.2.1 where $p_{0,B} + p_{0,HR} = 0.5$. Then $p_B = 1 - 0.9 \cdot p_{BR} - 0.9 \cdot p_{B,0} - p_{B,HR} - p_{B,HB}$. This decrease in p_{BR} and $p_{B,0}$ and the increases in p_B reflect Blue's better performance.

Figure 4.19 shows that the likely benefit of substantial investment in better platforms or payloads may only be marginal, as the technologically inferior side still wins in 25% of all cases regardless of investment level. Such technological enhancements are not only likely to be costly, the advantage is often also only temporary as the technology proliferates. A cost-benefit analysis can be performed as future work to more definitely address this trade space.

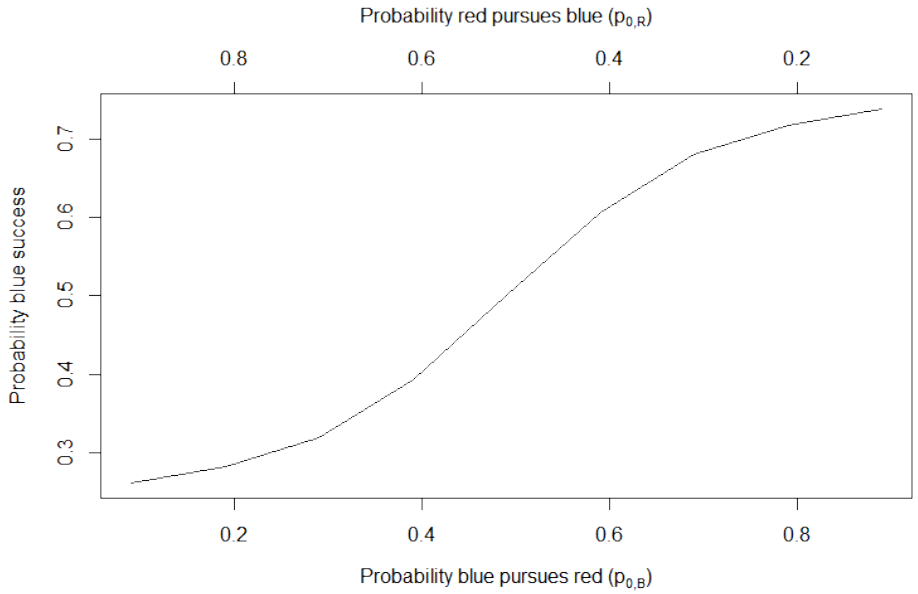


Figure 4.19: Blue success probability as function of $p_{0,B}$ and $p_{0,R}$. Even if one side has a large performance advantage, the opponent swarm still keeps a relatively high success probability.

Now we want to combine the former two analyses. We vary $p_{0,B}$ between 0.09 and 0.89 as before, while $p_{0,HR}$, $p_{B,HR}$ and $p_{R,HR}$ range from 0.0001 to 0.02. We observe the same general behavior in Figure 4.20 as in the analyses before. However, the results also indicate that adapted tactics can offset inferior performance. $p_{0,HR}$, $p_{B,HR}$ and $p_{R,HR}$ can be interpreted as the weight factor we defined for the simulation model. They determine the offensive and defensive behaviors of the swarm and is one part of swarm tactics. Even if Blue has a lower performance compared to the Red swarm, it is able to keep almost a 0.5 success rate by increasing the weight

factor. On the other hand, performance comprises not only hardware capabilities but also ”soft skills“ which include elements of swarm tactics. One important point is coordination among the swarm UAVs. For example, blind spots in the sphere around the swarm due to limited sensors could be mitigated by friendly UAVs actively fly to provide visibility of these known areas. Such additional coverage would increase the performance and offset a potential disadvantage. Based on these conclusions, we can observe that cost-intensive and time-consuming hardware changes may not only provide marginal improvement to the system, but that emphasis on coordination tactics for cooperating UAVs may provide even greater enhancements.

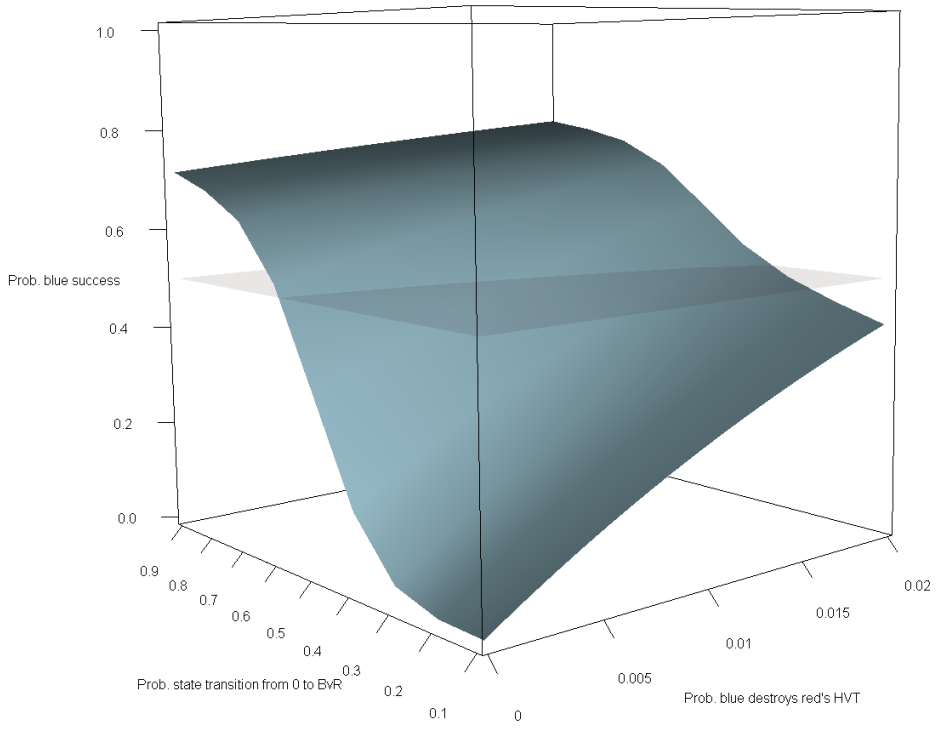


Figure 4.20: Blue success probability as function of $p_{0,B}$ and $p_{0,HR}$. The horizontal plane indicates the 0.5 success probability. A more offensive strategy can raise the success probability even though the performance is poor compared to the adverse swarm.

The analysis in this chapter identified important factors for UAV swarm tactics. For example, the weight factor governing offensive versus defensive preferences for the swarm turned out to be most significant for the defined scenario. Furthermore, we saw that the simulation and Markov process models lead to similar results which increases the confidence in the respective models. The advantage of the agent-based simulation model is the ability to investigate specific agent-level parameters and their impact on the swarm engagement, whereas the advantage of the Markov process formulation is that it enables sensitivity studies, including analysis of opposing

swarms of different capabilities (e.g., better platforms). Though both models can be enhanced to provide greater representation of a more realistic engagement, such as including draws, the insights provided in this thesis provide a basis for future recommendations and development of swarm systems and their tactics.

CHAPTER 5:
Conclusions and Future Work

5.1 Conclusions

The main objective for this thesis was to identify important factors that influence UAV swarm tactics. Though one could argue that such tactics development is not different from that for manned air combat, the basic conditions and engagement scenarios are not equivalent. The future warfare concept explored in this thesis addresses the employment of inexpensive and expendable platforms compared to their expensive and necessarily recoverable manned fighter aircraft counterparts. The loss of such a UAV also does not risk the life of a pilot. Further, it is conceivable that it could also be tactically beneficial to sacrifice UAVs in order to destroy an adversary's HVT. These tactics are surely different from combat where humans are directly involved. Moreover, we assume autonomous swarms with little to no direct control of individual UAVs by a ground station. In the ideal case, perhaps all the knowledge and understanding of an pilot should be implemented in the UAV. However, we know that this is technologically infeasible, and as such, we focused more on identifying the basic elements in such swarm UAV engagements that make the difference.

There are at least three approaches to address and analyze this novel future concept of swarm-versus-swarm UAV warfare. First, we could strive to find a solution through theoretical abstractions and apply historical cases to this new topic. Such an approach is likely a good point to start; however, the complexity of the scenario and the analytic assumptions will likely overlook important influences. The second approach is to conduct real-world experiments. However, given that this is an advanced warfare concept, we do not yet have the technological capabilities to perform operational-level experimentation. The third approach lies in between these two, and this hybrid perspective provides the desired capability assessment tools. We chose two different types of models, one being an agent-based simulation model and the other, a stochastic Markov process as an analytical model. Both have different foundations and our hope is that they complement one another.

We identified and here highlight six elements connected to the tactics of an UAV swarm, which merited study in this design and assessment effort, including:

- The maximum number of allocations per detected enemy target,

- The distance from the adversary's high value target at which the swarm starts to converge directly on this objective,
- The maximum distance relative to a UAV within which it recognizes swarm members for its cohesion behavior,
- A factor that describes a preference between offensive (i.e., attacking the HVT) and defensive (i.e., defending against attacking enemy UAVs) postures,
- The staying power of the high value target, and
- The initial spatial configuration of the deployed UAV swarm.

In leveraging the statistical design of experiments to systematically explore the factor space, we used a central composite design to scan for the importance of twelve identified variables and their interactions. The carefully constructed experimental design was executed, generating the following three response variables as measures of effectiveness:

- Blue success in destroying the Red HVT [yes/no]
- Number of surviving Blue UAVs
- Number of surviving Red UAVs

During the analysis portion of this thesis, we fitted a logistic regression model with Blue success as the response. The first key insight from this analysis is that all studied main effects are important. Either the main effects themselves are important or they are part of influential interactions. With an error rate of only about 14%, we developed a regression model that explains much of the variability in the simulation model, but allows for further study and sufficient uncertainty due to the "fog of war."

For the underlying scenario, it is important to destroy the opposing HVT before the adversary swarm reaches one's own HVT. The analysis showed that the outcome of the battle is significantly dependent on the Red swarm's choices of parameters under the assumptions of the given scenario. This effect could be due to the fact that, whereas the initial positions of the Blue swarm was varied substantially (to explore tactics involving one- or two-sided flank attacks or concentration of forces in the center), the initial positions of the Red swarm were always uniformly spatially distributed. The reduced variability of the Red swarm's initial positioning is attributed with this asymmetry in significance. Based on the analysis, a uniformly distributed initial configuration seems to be adaptable to most opposing situations, and overall appears to be the best choice if one does not know anything about the enemy's tactics.

The statistical analysis of the responses further highlighted specific insights with respect to several key swarm tactics attributes:

Weight factor: The analysis showed that the weight factors governing preferences for offensive versus defensive behaviors are most important to mission success, with the Red swarm's weight factor playing a significant role. The Red's value should be chosen high, such that the swarm primarily concentrates on the main objective of attacking Blue's HVT and nearly completely ignores the secondary objective of attacking incoming blue UAVs. The same relationship is true for the Blue swarm, but not to the same magnitude. The advantage of such a behavior is intuitive, since the swarm can accomplish the mission despite suffering many losses, and so there is not much benefit *to the mission* by attacking opposing UAVs in a defensive context. The analysis also showed that this weight depends (as an interaction) also on the staying power of the HVTs. For a higher staying power, more UAVs are needed to destroy the HVT, such that we observed that a lower weight factor is more appropriate (e.g., for staying power equal to eleven).

Convergent distance: Another important factor is **ConvergeRed**. Based on the analysis, the ideal choice for when attacking UAVs should begin converging on the HVT is approximately at the midpoint distance between the two HVTs. Recall that the experimental design we used consisted only of three levels per regressor, so the exact point cannot be determined. In contrast, **ConvergeBlue** as main effect does not appear to be very important, but we surmise that this fact is also a result of the variability in the initial positioning of the Blue swarm. A design that varies the Red positioning as well could explore this asymmetry.

Number of allocations: Interesting to note is the interaction between the weight factors and the number of allocations (i.e., the number of defending UAVs assigned to a single enemy UAV). First of all, since the weight factors are more important than the number of allocations (see above), we should choose the number of allocations according to the choice of the weight factor. The interactions show an inversely proportional positive relationship between the weight factor and number of allocations (i.e., a low weight factor benefits most from a high number of allocations and vice versa).

Initial positioning: Though hypothesized that initial deployment configurations would be important for influencing mission performance, the analysis showed that it was less important than expected for the given scenario setup. We could not find an initial positioning of the Blue swarm other than a centered and concentrated configuration that yielded favorable results. In

fact, additional runs with an optimized parameter choice derived from the analysis for initial positioning showed that such a concentrated and centrally-located configuration is superior to the uniform distributed setup of the red swarm.

Cohesion: This variable is also less important, which was surprising, given its role in dictating the collective swarm flight behaviors. More interestingly, the analysis suggests a loose swarm configuration with *less* influence between swarming agents. However, this result is mitigated by the choice of initial positioning, since all UAVs are nominally within each other's cohesive sphere of influence when the initial configuration is relatively compact and concentrated.

Staying power: It is not important on its own, but as previously mentioned, this variable's interaction with the weight factors does play a role. However, given that the staying power of the HVT is not one that is explicitly chosen by the swarm, this fact dictates that the swarm should choose its swarm tactics and parameters according to the given staying power of the swarm's assigned HVT.

The analysis of the Markov process model showed that the theoretical formulation points in the same directions as the simulation, which we had hoped to see, and assists in verifying the simulation model. In addition, we were able to relate the choice of transition probabilities in the Markov model to associated variables in the simulation model. As a result, we can leverage an analytical model, which can be used for macroscopic and on-the-fly analyses of swarm tactics, and rely on the constructed simulation model that provides a more detailed investigation but requires additional investment in conducting the analysis. The highlighted key insights into swarm tactics as well as the formulation of these two complementary models to explore them in detail are the main contributions of this thesis.

5.2 Future Work

We gained some insights about tactics in swarm versus swarm combat with the provided models. However, we have already identified several areas for improvement in the models themselves, in the analysis approaches, as well as in a deeper understanding of this topic of swarm tactics.

First of all, the generated regression models apply (and apply well) to the specific scenario studied in this thesis. However, the scenario is not of sufficient fidelity whereby the results could easily be transferred to swarm tactics employed in the real world. The biggest weakness is that the scenario stops (i.e., the simulation terminates) as soon as one HVT is destroyed. The analysis revealed that success of destroying the adversary's HVT and the resulting force ratio are

not unrelated. Rather, though one swarm was able to accomplish the mission, it lost so many UAVs that the remaining UAVs would not be able to protect the home base. Hence, both HVTs would be ultimately destroyed. This is a draw situation we do not presently allow in the current scenario. A dominant victory accomplishes both the main and secondary objective where a defeated UAV swarm is equivalent to a destroyed HVT. If we penalize a draw, perhaps analogous to draws in soccer, then the fitted model favors meeting the complete victory conditions. However, this modification also changes the analysis as the primary measure of effectiveness is no longer a binary response variable. Further, we must be cautious in applying the results found herein in a real combat situation, since the present study assumed that the Red swarm utilized the same objective function as the Blue swarm, which need not necessarily be true.

If we go with the last statement, then it may be beneficial to know how the adversary swarm behaves prior to the engagement. This highlights the advantage of performing good reconnaissance, for example, by dedicating a few of the UAVs to scouting missions. Another way to address uncertainty in the opponent's objective function could be to diversify and split the swarm into offense and defense components. Note, however, we saw the extreme cases where the friendly swarm is wholly defensive (weight factor is zero) and the adversary is offensive (weight factor is ten), in which case the offensive swarm has the advantage according to the interaction plot from the analysis. One solution to counter this issue could be to increase the distance between the home bases. Then the defense has more time to defeat the attacking swarm. Such line of reasoning, however, leads to additional questions, such as what is the ideal standoff distance? Though this could be another factor in future analysis, an increase in this distance between HVTs could have other operational considerations, especially when including endurance limitations and ammunition consumption in extended models.

After addressing tactics-related avenues for future work, we also see promising extensions to the simulation model formulation. We employ communication, sensors, and weapons systems under nearly perfect conditions in the model so far. However, in order to address uncertainty in warfare and nature, we could introduce probabilistic distributions to model real-world effects, for example. But does this really improve the model and substantially change the results? This last question is an interesting one that cannot be answered without further investigation.

The analytical model has also some potential for improvements. First of all, we constructed the transition probabilities, which were not based on any empirical observations neither from the real world nor from other models. Given the future concepts nature of this study, we cannot

get real world data; however, the simulation system could provide estimates of these values. However, since the states of the individual UAV agent in the simulation do not directly match the states in the Markov process, this latter approach may pose some challenges. Furthermore, the current Markov process model only allows one-on-one combat, but not multi-on-one. For sure, the number of possible states would grow exponentially, but short excursions with baseline two-on-one models, as found in the literature, could provide some insights. Another point for extension of the Markov model is the staying power of a HVT, which is constrained to be one, that is, the HVT can only sustain one hit. This limitation could be removed so as to further compare its results with those of the simulation model. Another way to model the scenario with an analytical approach is the use of Lanchester equations, which notably were originally developed to explore air-to-air combat. Though the resolution of this differential equation-based approach does not enable study of agent-to-agent interactions, it would be interesting to see how the results compare to the Markov process and the simulation model.

Finally, the scenario we study assumes that there are two swarms of UAVs that have a sense of cohesion, share knowledge of detected targets, and smoothly negotiate target allocations. But an open question remains as to whether these capabilities really matter? What if one swarm does not have one or more of these capabilities? Such an interesting question could be straightforwardly answered with the presented simulation model in near-term future work. Other modifications of the scenario could include jamming, which reduces the communication capabilities among the UAVs, or the challenge of unknown HVT positions, in which swarm search algorithms would be needed.

Overall, this thesis provides a first step into the complex world of swarm-versus-swarm tactics. Both simulation and theoretical models provide a basis for significant follow-on research. In actuality, this thesis could not provide solid answers that are immediately transferable to the operational world. However, the value and contributions of this thesis are that it has prepared the stage with two tools, by sketching a way through the whole process and by identifying additional questions that are worthwhile to addressing.

APPENDIX A:
Simulation Environment Installation

Run the JARs

We provide three *jar*-files (Java Archive) with this thesis. All three can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

- `UAVSwarm_source.jar` comprises the whole project with *java* and *class* files.
- `UAVSwarm.jar` is executable and runs one of the provided CSV-files with the generated DoE (see Section 3.2). The output is a CSV-file as explained in Section 3.4. You are asked to provide an appropriate CSV-file from your file system. After the run you can select a location and a filename for the simulation output.
- `UAVSwarmGUI.jar` is also executable and runs the first design points in the *csv*-files. You have to install the Java3D and Java Media Framework (JMF) package as explained in the next section in order to run this JAR.

Setup MASON in Eclipse to Run Your Own Model

The guidance in this section comprises all necessary software modules to setup the Java project we developed during this thesis. The steps are almost the same for all available platforms but they differ in the details. Here we only show how it works for a Windows machine as this is still the most common operating system. The mentioned file locations will differ according to your operating system.

Initial Steps

First, of all you must have a Java runtime environment (JRE) installed. Every JRE above Java 6 should work in this case. We recommend Eclipse (`http://www.eclipse.org/`) as the development environment because this is a free, well supported, and commonly used tool among Java developers. The software can be downloaded from the provided link. Simply unzip and run `eclipse.exe`, though we recommend to coping the folder to your `Program Files` directory and adding a shortcut into your *Start* menu.

Then you need some additional packages:

- Install Java3D, ensuring the right package for your computer architecture (32- or 64-bit) (`http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html`). After installation make sure that the path `C:\ProgramFiles\Java\Java3D\1.5.1\bin\` is included in your system's *Environmental Variables* under *Path*. Otherwise, *j3dcore-ogl.dll* cannot be found.
- Download and install the Java Media Framework (`http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html`). We recommend *(*.exe) file.
- Download and unzip *Super CSV* (`http://supercsv.sourceforge.net/`). This gives you support to handle CSV-files.
- Download and unzip MASON (`http://cs.gmu.edu/~eclab/projects/mason/`).

Setup MASON

First you should *import* MASON as a new project in Eclipse.

The second step is to reference the Java3D JARs (`j3dcore.jar`, `j3dutils.jar`, `vecmath.jar`) to the project (*Project->Properties->Java Build Path->Libaries*). They can usually be found in `C:\ProgramFiles\Java\Java3D\1.5.1\lib\ext\`. Do the same for the JMF JARs in `C:\ProgramFiles(x86)\JMF2.1.1e\lib\`.

To confirm this is working, run one of the 3D applications in the *sim.app* package. In order to see the graphical output, run the `...WithUI.java` class as a *Java Application*.

Get the UAVSwarm Project Running

Import `UAVSwarm_source.jar` as *Existing Project* into your workspace. Go again to *Project->Properties->Java Build Path->Libaries* and reference the Java3D and JMF JARs for this project. Then go to the *Projects* tab and add MASON as the required project.

Test this configuration by running the `UAVGUI.java` class.

APPENDIX B:
Complete list of fitted $\hat{\beta}$s

This appendix shows all terms and the corresponding coefficients of the final logistic regression model in descending order. The magnitude of the coefficient is a measurement of importance. The most significant ones are discussed in Section 4.1.4.

Term	$\hat{\beta}$
weightRed	-2.40700031
convergeRed	-2.34268830
I(convergeRed^2)	2.02622621
I(weightBlue^2)	1.75709594
weightRed:rhs(DistanceCenter, 3000)	-1.63271293
numAllocBlue:weightBlue	-1.30740299
numAllocRed:weightRed	1.08292601
weightRed:stayingPwHVT	1.04541938
weightBlue:stayingPwHVT	-0.97929942
weightBlue	0.82523436
numAllocRed:stayingPwHVT	-0.79081038
weightBlue:weightRed	-0.70730609
convergeBlue:weightBlue	0.69240074
numAllocBlue	0.68324259
convergeRed:weightRed	-0.66977778
DistanceUAV	-0.65948801
weightRed:DistanceUAV	-0.61042046
cohesionMax:weightBlue	-0.57027147
numAllocBlue:stayingPwHVT	0.55291950
weightBlue:rhs(DistanceCenter, 3000)	-0.47040495
cohesionMax:DistanceUAV	0.40974145
numAllocRed:weightBlue	-0.40864333

continued on next page

continued from previous page

Term	$\hat{\beta}$
cohesionMax:weightRed	0.39484181
cohesionMax	-0.37815030
lhs(DistanceCenter, 1500)	0.36875794
numAllocBlue:rhs(DistanceCenter, 3000)	0.34334688
numAllocRed	-0.33312201
convergeBlue:lhs(DistanceCenter, 1500)	-0.27138983
convergeBlue:weightRed	0.26469812
convergeBlue	0.24650393
weightBlue:DistanceUAV	-0.23308254
center(DistanceCenter, 1500, 3000)	0.20957125
weightRed:center(DistanceCenter, 1500, 3000)	-0.20468967
numAllocBlue:weightRed	0.19752579
numAllocRed:convergeRed	0.18631218
convergeBlue:center(DistanceCenter, 1500, 3000)	-0.17804176
weightRed:lhs(DistanceCenter, 1500)	0.17293492
numAllocBlue:DistanceUAV	0.17009838
convergeBlue:convergeRed	0.16355295
cohesionMax:stayingPwHVT	0.15406344
I(weightRed^2)	0.14528475
convergeRed:rhs(DistanceCenter, 3000)	-0.14026918
numAllocRed:convergeBlue	-0.13515476
stayingPwHVT:DistanceUAV	0.12952884
numAllocBlue:numAllocRed	0.12101517
cohesionMax:lhs(DistanceCenter, 1500)	-0.11264724
convergeRed:stayingPwHVT	0.10451249
weightBlue:lhs(DistanceCenter, 1500)	0.10080354
numAllocBlue:center(DistanceCenter, 1500, 3000)	0.09945470
numAllocRed:rhs(DistanceCenter, 3000)	-0.08921660
rhs(DistanceCenter, 3000)	-0.08806592
numAllocRed:center(DistanceCenter, 1500, 3000)	-0.07276992

continued on next page

continued from previous page

Term	$\hat{\beta}$
stayingPwHVT:lhs(DistanceCenter, 1500)	-0.07254442
numAllocRed:cohesionMax	-0.07117946
numAllocBlue:convergeRed	-0.06902416
convergeBlue:DistanceUAV	0.06282924
cohesionMax:center(DistanceCenter, 1500, 3000)	-0.06238867
cohesionMax:rhs(DistanceCenter, 3000)	0.06003309
numAllocBlue:cohesionMax	0.05982500
convergeBlue:rhs(DistanceCenter, 3000)	-0.05889753
DistanceUAV:rhs(DistanceCenter, 3000)	0.05888981
cohesionMax:convergeRed	-0.05425259
numAllocRed:DistanceUAV	0.05360961
numAllocRed:lhs(DistanceCenter, 1500)	-0.05287791
numAllocBlue:convergeBlue	-0.05039057
weightBlue:center(DistanceCenter, 1500, 3000)	0.05022147
stayingPwHVT:center(DistanceCenter, 1500, 3000)	-0.04688542
convergeRed:center(DistanceCenter, 1500, 3000)	-0.04247425
convergeRed:weightBlue	-0.04041143
DistanceUAV:lhs(DistanceCenter, 1500)	-0.04040390
DistanceUAV:center(DistanceCenter, 1500, 3000)	0.02971351
cohesionMax:convergeBlue	0.02934506
convergeRed:DistanceUAV	0.02626016
convergeBlue:stayingPwHVT	-0.02441405
convergeRed:lhs(DistanceCenter, 1500)	-0.01687805

Table B.1: Sorted terms of the fitted model with $\hat{\beta}$s.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] H. Geer and C. Bolkcom, "Unmanned aerial vehicles: Background and issues for Congress," *CRS Report for Congress*, CRS Report No. RL31872.

[2] P. Feigin, O. Pinkas, and J. Shinar, "A simple Markov model for the analysis of multiple air combat," *Naval Research Logistics Quarterly, vol. 31*, 1984.

[3] W. Nunn and R. Oberle, "Evaluating air combat maneuvering Engagements Vol. I - methodology," *Center for Naval Analyses CNS*, vol. I, no. Op 03, 1976.

[4] D. Frelinger, J. Kvitky, and W. Stanley, "Proliferated autonomous weapons," Rand Corp., Santa Monica, CA, Tech. Rep., 1997.

[5] B. T. Clough, "UAV swarming? So what are those swarms, what are the implications, and how do we handle them?" in *AUVSI Unmanned System Conf.*, Orlando, FL, 2002.

[6] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, Aug. 1987.

[7] I. C. Price, "Evolving self-organized behavior for homogeneous and heterogeneous UAV or UCAV swarms," Ph.D. dissertation, Air Force Air University, Wright-Patterson Air Force Base, OH, 2006.

[8] M. Day, "Multi-agent task negotiation among UAVs to defend against swarm attacks," M.S. thesis, Naval Postgraduate School, Monterey, CA, 2012.

[9] B. J. Cobb, "Adaptive discrete event simulation for analysis of Harpy swarm attack," M.S. thesis, Naval Postgraduate School, Monterey, CA, 2011.

[10] Barbara Starr, "Iranian jets fire on U.S. drone," 2012. [Online]. Available: http://security.blogs.cnn.com/2012/11/08/first-on-cnn-iranian-jets-fire-on-u-s-drone/

[11] CNNWireStaff, "Israel shoots down drone," 2012. [Online]. Available: http://www.cnn.com/2012/10/06/world/meast/israel-drone-downed/index.html

[12] ArmyUASCoEStaff, "Eyes of the Army–US Army Roadmap for Unmanned Aircraft Systems 2010–2035," *US Army UAS Center of Excellence*, 2010.

[13] R. Shaw, *Fighter combat: Tactics and maneuvering*. Annapolis, MD: USNI, 1985.

[14] S. Edwards, *Swarming on the Battlefield: Past, Present, and Future*. Santa Monica, CA: Rand Corp., 2000.

[15] T. W. Lucas and S. M. Sanchez, "Exploring the world of agent-based simulations: simple models, complex analyses," in *Proc. of the 2002 Winter Simulation Conf.* San Diego, CA: IEEE, 2002, pp. 116–126.

[16] D. Nowak, "Exploitation of self organization in uav swarms for optimization in combat environments," Ph.D. dissertation, Air Force Air University, Wright-Patterson Air Force Base, OH, 2008.

[17] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based simulation platforms: review and development recommendations," *Simulation*, vol. 82, no. 9, pp. 609–623, Sep. 2006.

[18] George Mason University, "MASON," 2013. [Online]. Available: http://cs.gmu.edu/~eclab/projects/mason/

[19] T. Cioppa, T. W. Lucas, and S. M. Sanchez, "Military applications of agent-based simulations," *Proc. of the 2004 Winter Simulation Conf., 2004.*, vol. 1, pp. 165–174, 2004.

[20] S. Luke, C. Cioffi-Revilla, and L. Panait, "MASON: A multiagent simulation environment," *Simulation*, pp. 1–18, 2005.

[21] The Eclipse Foundation, "Eclipse," 2013. [Online]. Available: http://www.eclipse.org/

[22] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer. J. of Math.*, vol. 79, no. 3, pp. 497–516, 1957.

[23] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a Dubins airplane," in *2007 46th IEEE Conf. on Decision and Control*. New Orleans, LA: Ieee, 2007, pp. 2379–2384.

[24] S. M. Ross, *Introduction to Probability Models, Tenth Edition*. Burlington, MA: Academic Press, 2009.

[25] S. M. Sanchez, "Work Smarter, Not Harder: Guidelines for Designing Simulation Experiments," *in Proc. of the Winter Simulation Conf., 2005.*, pp. 69–82, 2005.

[26] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa, "State-of-the-art review: A user's guide to the brave new world of designing simulation experiments," *INFORMS J. on Computing*, vol. 17, no. 3, pp. 263–289, Jul. 2005.

[27] N. Draper and D. Lin, "Capacity considerations for two-level fractional factorial designs," *J. of statistical planning and inference*, 1990.

[28] P. J. Sanchez, "Efficient generation of resolution VII fractional factorial designs," unpublished.

[29] S. M. Sanchez, "NOLH designs spreadsheet," 2011. [Online]. Available: http://harvest.nps.edu/

[30] D. D. Wackerly, W. Mendenhall III, and R. L. Scheaffer, *Mathematical Statisctics with Applications*, 7th ed. Belmont, CA: Books/Cole, Cengage Learning, 2008.

[31] A. Law, *Simulation Modeling and Analysis with Expertfit Software*. New York, NY: McGraw-Hill Science/Engineering/Math, 2006.

[32] J. J. Faraway, *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models (Chapman & Hall/CRC Texts in Statistical Science)*. Boca Raton, FL: Chapman and Hall/CRC, 2005.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. New York: Springer, 2009.

[34] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. New York: Wiley, 2010.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California
2. Defense Technical Information Center
   Ft. Belvoir, Virginia